5-6-2014

Versie 1.0

Afstudeerverslag

Social media plug-in module

- **School**: De Haagse Hogeschool
- **Opleiding**: Informatica
- **Docenten**:
  - G.M. Tuk
  - J.D. Maas

Stijn van der Meer, 10006877
WEBBEAT

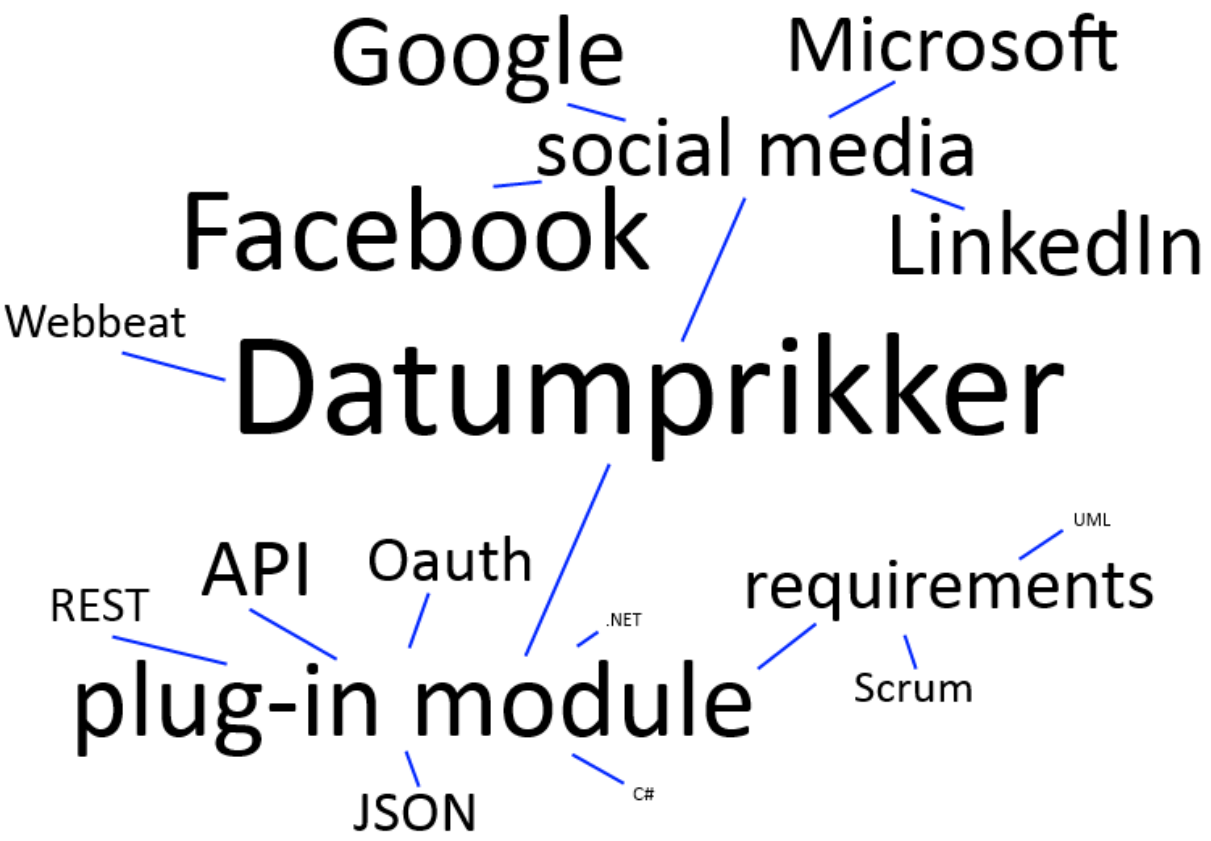
Voorwoord

Dit verslag is gemaakt voor het project “social media plug-in module”, dat in opdracht van Webbeat is uitgevoerd door mij als afstudeerder bij de Haagse Hogeschool. Ik wil daarom dan ook de medewerkers van Webbeat bedanken voor hun medewerking, begeleiding en hulp bij het realiseren van het eindproduct.

Referaat

Het ontwerpen, bouwen en testen van een (generieke) plug-in module voor social media.

keywords				
.NET	API	C#	Datumprikker	Facebook
Google	JSON	LinkedIn	Microsoft	Oauth
plug-in module	Requirements	REST	Scrum	social media
UML	Webbeat			



Inhoudsopgave

Leeswijzer

Dit document is opgebouwd uit 4 onderdelen:

- Inleiding; hierin wordt het project uitgelegd en de voorbereiding hierop beschreven.
- Uitvoering; hierin worden de vijf sprints en de uitgevoerde testen chronologisch beschreven.
- Evaluatie; hierin wordt het hele project geëvalueerd, en de competenties beschreven.
- Bijlages; hierin staan alle bijlages die bij dit document horen. De (tussen)producten die in de loop van het project zijn opgeleverd zijn samengevoegd in één extern document.

Dit document houdt de volgende tekstopmaak voor de tekst aan:

- **Document**; dit is een verwijzing naar een document binnen het project.
- **Hoofdstuk**; dit is een verwijzing naar een hoofdstuk of paragraaf. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar het betreffende hoofdstuk(of paragraaf) brengen.
- **Afbeelding**; dit is een verwijzing naar een afbeelding. In de digitale versie van dit document zal deze verwijzing aanklikbaar zijn, en naar de betreffende afbeelding brengen.
- **Variabele**; dit is een verwijzing naar een variabele die gebruikt wordt binnen een bepaalde context (bijvoorbeeld binnen een **afbeelding**).
- **Package**; dit is een **variabele** dat een package voorstelt. De reden dat er hier onderscheid gemaakt wordt, is omdat er in dit document vaak een package variabele wordt gebruikt. Daarnaast bevat een package (vaak) andere variabelen.

Dit document houdt verschillende termen aan. Betekenissen en/of uitleg van deze termen staan vermeld in bijlage 2: Begrippenlijst.

1 Inleiding

Webbeat Products B.V. (Webbeat) is opgericht in 2000 in Den Haag. Nu gevestigd in Wateringen, is Webbeat sinds 2012 onderdeel geworden van Nalta Group te Almere. Samen bieden zij een complete dienstverlening voor projecten op het gebied van hardware en software. Webbeat heeft op het moment van schrijven acht medewerkers in dienst die onder een platte organisatiestructuur samenwerken aan verschillende projecten.

Hierbij wordt er gebruik gemaakt van een zelfontwikkelde lichte vorm van Scrum als software ontwikkelmethode, waarbij (interne) communicatie en overleg centraal staat. Doordat de organisatie relatief klein is qua omvang, is het snel en wendbaar. De medewerkers werken vaak in kleine groepjes of in hun eentje aan verschillende opdrachten.

Webbeat is gespecialiseerd in het ontwikkelen van web applicaties op basis van Microsoft technologie. Daarnaast worden er ook app's ontwikkeld voor de meest gangbare mobiele platformen (iOS, Android, Windows Phone 8). De meeste projecten worden in opdracht van andere bedrijven uitgevoerd, maar Webbeat heeft ook een aantal eigen producten ontwikkeld die worden aangeboden als online dienst (SAAS). Een van deze producten is Datumprikker, waarmee op een snelle en eenvoudige wijze een geschikte datum kan worden bepaald voor een afspraak. Ooit begonnen als een kleine tool, is Datumprikker inmiddels gegroeid naar een serieuze applicatie waar meer dan een miljoen unieke bezoekers per maand gebruik van maken.

Deze gebruikers hebben uiteraard allerlei wensen, en daarom wordt er dan ook continu gekeken of deze wensen verwezenlijkt kunnen worden. Eén van deze wensen is om social media platforms (SmP's) te kunnen gebruiken bij Datumprikker. Hierbij moet er dan gedacht worden aan:

- Inloggen op Datumprikker met het account van een SmP.
- Het uitnodigen van contacten die de gebruiker kent bij het SmP.
- Berichten sturen naar deze contacten.
- Controleren of een datum al bezet is in de kalender van het SmP.
- Indien een datum gekozen is, deze (automatisch) in de kalender van het SmP zetten.

1.1 Opdracht

De wensen die hierboven staan, zouden, met behulp van de afstudeeropdracht, daadwerkelijke functionaliteiten moeten gaan worden. Dit zou verwezenlijkt kunnen worden door voor elk SmP de library of SDK te downloaden, en dan vervolgens de functionaliteiten één voor één te gaan bouwen. Hierdoor zou er dan echter iedere keer een soortgelijke functionaliteit gebouwd worden, zonder deze met elkaar te koppelen.

Een betere oplossing zou zijn om een eigen module te bouwen, die dan per SmP een eigen plug-in bevat. Deze plug-ins kunnen dan gemakkelijk toegevoegd, gewijzigd en/of verwijderd worden. De reden dat het bouwen van een (eigen) module beter is, is goed uit te leggen aan de hand van de ISO 25010 norm[1] (vernieuwde versie van ISO 9126):

- **Functional suitability**: Bij een zelfgemaakte module wordt er alleen aan de benodigde behoeften voldaan. Hierdoor verberg je onnodige functionaliteit.
- **Performance efficiency**: Hoewel een SDK/library mogelijk sneller is dan een eigen module, zijn er een hoop extra resources (**Resource utilization**) nodig om al die SDK'S/libraries te kunnen gebruiken. Een eigen module hergebruikt een hoop van die resources, en is daardoor lichter, zonder de benodigde functionaliteit op te hoeven geven.
- **Compatibility**: Door het gebruik van een zelfgemaakte module, worden alle resources en functionaliteit bij 1 punt gelegd (de module zelf). Hierdoor is het veel gemakkelijker om dit over te zetten naar een ander product, dan wanneer de individuele SDK's/libraries overgezet (en aangepast) moeten worden.
- **Usability**: Door het gebruik van een zelfgemaakte module, wordt er globaal vastgelegd wat de toegankelijkheid (**accessibility**) is voor de gebruiker. Hierdoor hoeft de UI (**User interface aesthetics**) niet los aangepast te worden per SmA.
- **Reliability**: Een zelfgemaakte module is betrouwbaarder dan een collectie van SDK's/libraries omdat er veel minder externe invloed is. Hierdoor zijn de **Availability**, **Fault tolerance** en de **Recoverability** beter beheersbaar.
- **Security**: Een eigen module is niet automatisch veiliger dan het gebruik van de SDK's/libraries. Maar doordat er maar één toegangspunt is in plaats van meerdere, is de verantwoordelijkheid (**Accountability**) en het authentiseren(**Authenticity**) wel makkelijker te beheren.
- **Maintainability**: Een zelfgemaakte module is veel makkelijker te onderhouden omdat het om één module (**Modularity**) gaat, die makkelijk te wijzigen (**Modifiability**), hergebruiken (**Reusability**), analyseren (**Analyzability**) en te testen (**Testability**) is.

[1] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

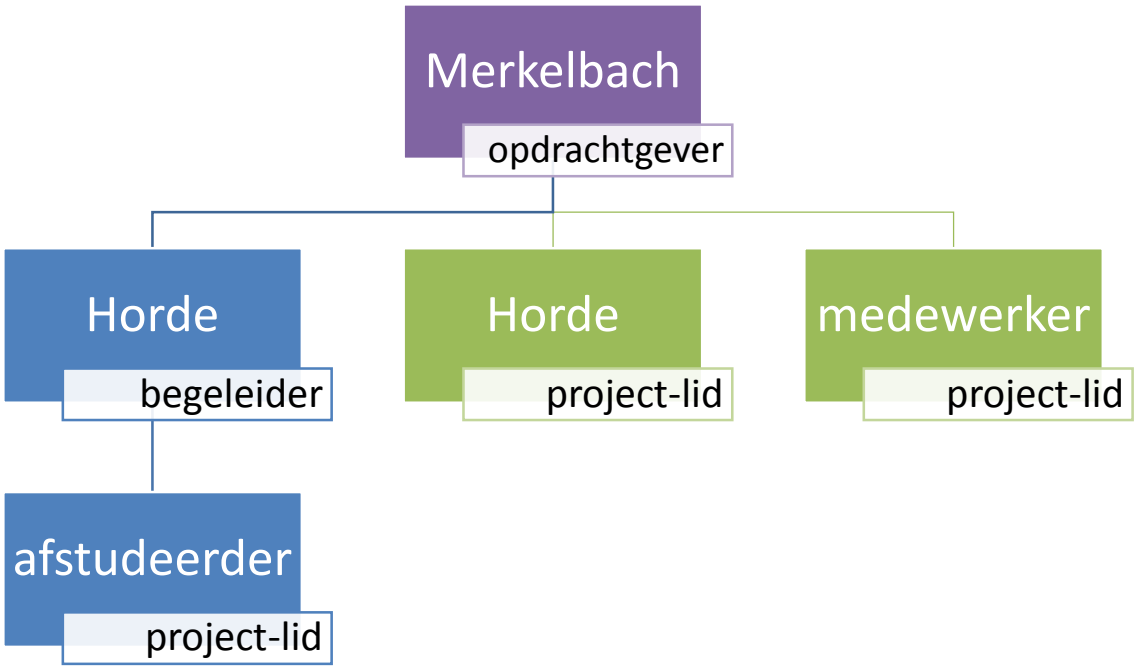
- **Portability**: Een eigen gemaakte module kan veel makkelijker overgezet worden naar een ander product/platform, doordat alle eisen en functionaliteit op één punt zitten. Daarnaast kunnen er makkelijker wijzigingen(**Adaptability**) doorgevoerd worden.

Om de plug-in module te kunnen vullen met de social plug-ins, moest er uiteraard eerst onderzocht worden welke social media er interessant zijn voor Datumprikker. Op basis daarvan konden dan de requirements opgesteld worden, en vervolgens het daarop gebaseerde ontwerp.

De social media plug-in module zou echter niet de enige wijziging in Datumprikker worden binnen de afstudeerperiode. Webbeat was van plan om een grote update te maken voor Datumprikker, met verschillende nieuwe functionaliteiten.

1.2 Project groep

Omdat de plug-in module een project was binnen het Datumprikker project, hield dit in dat er twee projectgroepen waren, die overlapping met elkaar hebben. In het onderstaande organigram is dit duidelijk te zien.



Figuur 1, organigram projectgroep

De blauwe blokken in **Figuur 1** staan voor het afstudeerproject, en de groene voor Datumprikker. Bij beide groepen is het paarse blok de opdrachtgever.

Voor dit project waren er dus drie rollen, en drie personen. Het grootste gedeelte van het project zou echter (zelfstandig) door het project-lid uitgevoerd worden.

1.3 Omgeving

Aangezien Datumprikker een .NET product is, zal de plug-in module ook in .NET gebouwd worden. Dit houdt in dat de meeste code in C# gebouwd zal worden.

Omdat de module binnen een ander project valt, werd er besloten om een kopie van die omgeving te geven. Deze kopie werd binnen een eigen ontwikkelomgeving gezet, zodat er aan de module gewerkt kon worden, zonder dat dit invloed had op het hoofdproject.

1.4 Stakeholders

Voor dit project waren er twee stakeholders:

- Opdrachtgever; de opdrachtgever wilde een nieuwe set van functionaliteiten binnen Datumprikker. Hij had dan ook grote belangen bij het slagen van dit project.
- Gebruiker; de gebruikers wilden graag gebruik maken van de nieuwe functionaliteiten. Voor deze groep werd het project dan ook opgezet. Indien deze groep niet tevreden gemaakt zou worden, zouden deze (mogelijk) naar concurrerende partijen overstappen.
- Afstudeerder; aangezien het afstudeercijfer afhangt van de kwaliteit van het project en diens producten, heeft de afstudeerder uiteraard veel belang bij dit project.

Naast de stakeholders waren er verschillende partijen die invloed hebben op het project en/of product, maar die geen (grote belangen) hadden bij het slagen van het project:

- Social media; aangezien er gebruik gemaakt zou worden van de producten van deze partijen, waren er hier eisen aan verbonden. Met deze eisen moest rekening gehouden worden.
- Webbeat; de medewerkers van Webbeat waren met een eigen project bezig voor Datumprikker. Aangezien het afstudeerproject hieronder viel, moest er rekening gehouden worden met de eisen en veranderingen die hierbij hoorden.

1.5 Plan

Periode:

Voorbereiding															
week	1					2					3				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

1.5.1 Software ontwikkelmethode

Om er voor te zorgen dat de communicatie tussen Webbeat en de afstudeerder goed kon verlopen, was het verstandig om de werkwijze van Webbeat en die van de afstudeerder zo veel mogelijk overeen te laten komen.

De medewerkers van Webbeat houden als software ontwikkelmethode een (eigen) lichte vorm van Scrum aan, waarbij de focus ligt op de (software) resultaten, en niet op de documentatie. Daarnaast bestaat Webbeat niet uit tientallen medewerkers, waardoor het vaak voorkomt dat de werkgroepen uit twee of zelfs maar één persoon bestaan. Communicatie staat echter wel centraal, dus de werkgroepen overleggen regelmatig met elkaar over de verschillende projecten. Dit gebeurt op een ongestructureerde manier. Dat wil zeggen: er zijn geen vaste momenten waarop er om advies en/of voortgang gevraagd wordt.

Doordat er op elk moment overlegd kan worden en er dus wijzigingen gemaakt kunnen worden in de requirements en/of ontwerp, worden deze voornamelijk mondeling vastgelegd, dan wel globaal opgeschreven. Dit alles bevoordeeld de mogelijkheid om snel software te kunnen ontwikkelen, en hierop controle en/of wijzigingen uit te voeren. Het is echter geen goede methode voor een afstudeerproject, aangezien documentatie hier wel centraal staat.

Om de vrije communicatie aan te kunnen blijven houden, en daarnaast de normale gang van zaken van Webbeat zo min mogelijk te storen, was het verstandig om deze methode zo veel mogelijk aan te houden. Ten behoeve van het afstudeertraject was het echter wel nodig om de documentatie aan deze methode (terug) in te voeren. Deze documentatie bestaat uit:

- Plan van aanpak (**1.5.2**); hierin staat in detail uitgeschreven hoe het project uitgevoerd zal worden.
- (voor)Onderzoek (**1.5.3**); hierin staan alle resultaten van de uitgevoerde onderzoeken.
- Requirements (**1.5.4**); hierin staan alle eisen waaraan de plug-in module moet voldoen.
- Ontwerp (**1.5.5**); hierin staat beschreven hoe de plug-in module eruit komt te zien.
- Test rapportage[2]; hierin staan alle tests en de bijbehorende resultaten beschreven.

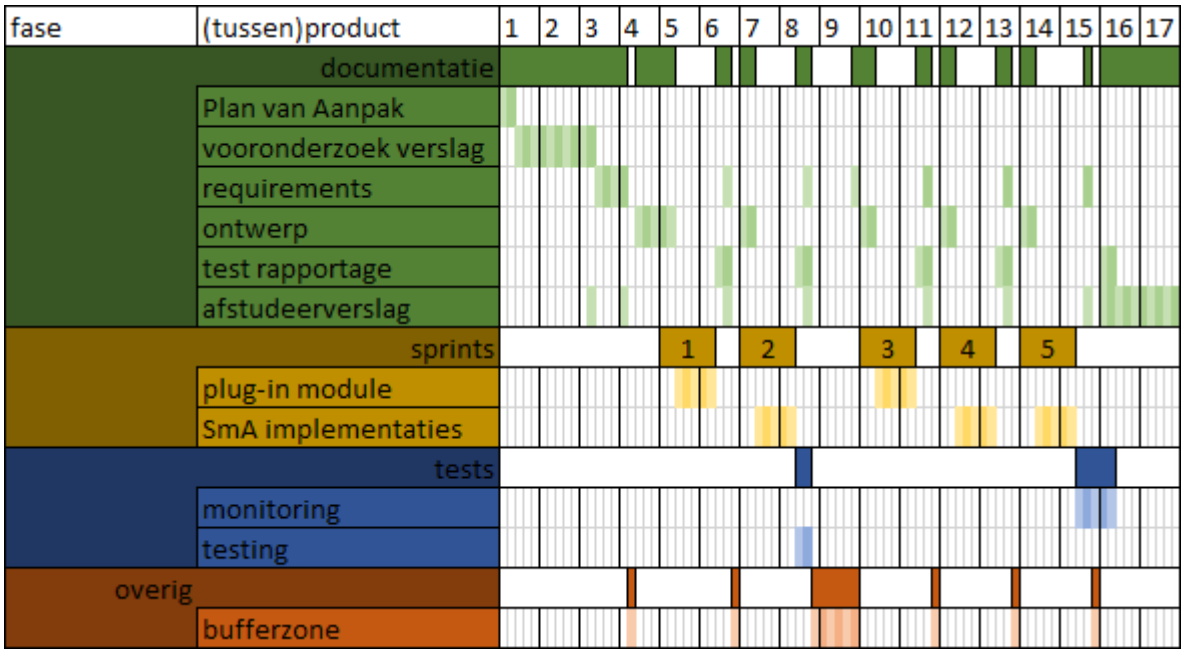
De meeste van deze documenten werden (grotendeels) opgesteld voordat er met de sprints begonnen werd. De initiële documenten worden hieronder kort beschreven. Tijdens de sprints zijn er aanpassingen gemaakt op deze documenten.

1.5.2 Plan van aanpak

Een groot deel van de onderdelen uit dit document konden gehaald worden uit het **afstudplan_2014-1.1 10006877** document. Daarnaast zijn de volgende dingen toegevoegd:

- **Software ontwikkelmethode**; Hierin wordt dus in detail uitgelegd hoe de Scrum variant voor dit project uitgevoerd zal worden.
- **Stakeholders**; Er moest een duidelijke scheiding gemaakt worden tussen de belanghebbende partijen (Webbeat, gebruikers), en partijen die wel eisen oplegde (SmP's), maar die geen (significant) belang hadden bij het slagen van het project.
- **Scope**; Omdat het project binnen een groter project uitgevoerd werd, moesten er duidelijke afspraken gemaakt worden over wat wel, en wat niet binnen de scope van dit project viel.
- **Planning**; Hoewel het lastig is om een concrete planning te maken bij een Scrum(variant) project, was het wel van groot belang dat er rekening gehouden werd met de deadline van het afstudeertraject. Vandaar ook dat er een globale planning opgesteld werd, die als blauwdruk kon fungeren voor de planning bij de sprints.

[2] Dit document is tijdens de sprints aangemaakt, niet tijdens de voorbereiding.



Figuur 2, originele planning

Wat opvalt bij **Figuur 2**, is dat er vier onderdelen voor de sprints zijn. De eerste twee zijn de twee typen sprints: de module, en de implementaties. Daarnaast zijn **monitoring** en **testing** meegenomen om te kunnen testen of de module werkt volgens de requirements.

Het globale idee was dus om (na de voorbereiding) eerst de eerste versie van de module te maken, en daarna de eerste implementatie. Hierna zou er een testfase volgen, met daarbij meteen de volgende sprint die de aanpassingen zou doorvoeren. Als laatste zouden de overige implementaties doorgevoerd worden. Aangezien deze gebaseerd worden op de vorige implementaties, zou dit geen (grote) wijziging betekenen op de module. Om te controleren of uiteindelijk alles naar behoren werkt, zou er een monitoring sessie plaats gaan vinden. Hierin zouden eventuele onvoorziene problemen gedetecteerd en opgelost worden.

1.5.3 Vooronderzoek

Voordat er iets gedaan kon worden voor de plug-in module, moest er eerst achterhaald worden wat er precies nodig was. De SmP's hebben namelijk allerlei eisen en/of beperkingen die voor de plug-ins zouden gelden. Bij dit onderzoek is er gekeken naar de volgende SmP's:

- Facebook
- LinkedIn
- Google(PLUS)
- Microsoft (Office365 en Live/Hotmail)

De volgende punten kwamen bij het onderzoek naar voren:

- Er moest voor elke SmP een account aangemaakt worden om gebruik te kunnen maken van diens social media API's (SmA's).
- De plug-in module moest met de volgende methodes/protocollen omgaan:
    - OAuth; versie 2 was bij alle onderzochte SmP's de standaard.
    - REST; een manier van communiceren tussen 2 omgevingen.
    - JSON; een manier om data uit te wisselen tussen 2 omgevingen.

Een belangrijke conclusie die getrokken kon worden uit het onderzoek, was dat de deze SmP's op veel vlakken dezelfde eisen en/of beperkingen hadden. Hierdoor waren er veel minder eisen dan eerst gedacht werd.

Op basis van de resultaten van dit onderzoek, bepaalde de opdrachtgever dat Microsoft Office 365 niet gebruikt zal worden voor dit project. Deze keuze werd bepaald door het feit dat dit platform vooral gericht is op bedrijven, en het authentiseren ook op basis gebeurd van deze bedrijven, en niet op basis van gebruikers.

1.5.4 Initiële requirements

Omdat de eisen die uit het vooronderzoek naar voren kwamen, veel minder waren dan eerst gedacht, had dit ook een impact op de hoeveelheid requirements die hiervoor opgesteld moesten worden. Wel kwamen er al een aantal andere requirements naar voren tijdens de eerste meetings met de opdrachtgever.

Hoewel de hoeveelheid eisen meeviel, was het wel lastig om al deze eisen van de verschillende SmP's met elkaar te vergelijken en verifiëren. Dit kwam doordat de informatie over de eisen per SmP vaak verspreid stond over (veel) verschillende bronnen. Daarnaast waren er soms verschillende versies en/of SmA's beschikbaar bij de SmP met elk zijn eigen eisen en/of beperkingen.

Om er voor te zorgen dat de requirements correct en duidelijk waren/bleven, is, net zoals bij het maken van de keuze voor de plug-in module in de paragraaf **1.1**, de ISO 25010[3] norm aangehouden bij het opstellen van de requirements.

[3] http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=35733

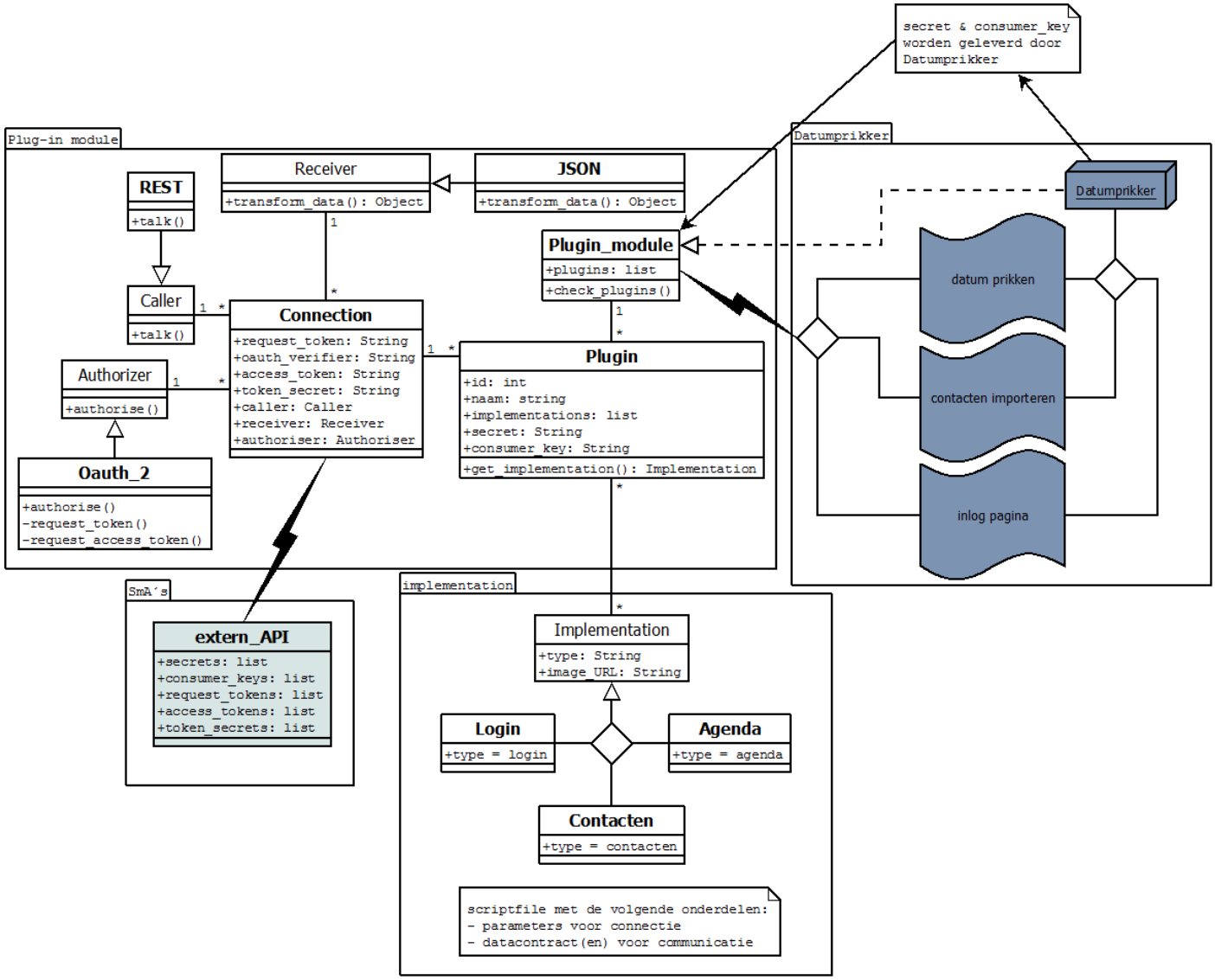
1.5.5 Initieel ontwerp

Nadat de eerste requirements opgesteld waren, kon er een eerste ontwerp gemaakt worden. Hierbij is er rekening gehouden met de volgende aspecten:

- Er moet gecommuniceerd, geauthentiseerd en geconverteerd worden met de (data van de) SmA's.
- Zowel de Datumprikker webpagina's als de applicatie zelf hebben een (communicerend) verband met de plug-in module.
- Het moet mogelijk zijn om (in de toekomst) meerdere types te hebben van:
  - **Authorizer**
  - **Caller**
  - **Receiver**
  - **Implementation**

Op basis van deze aspecten en de (overeenkomende) requirements, is een ontwerp (te zien in **Figuur 3**) opgesteld. Er is bij dit ontwerp een goede scheiding te zien tussen de 4 elementen die bij dit project van belang zijn:

- **Plug-in module**; dit omvat alle klassen die voor de plug-in module zelf van belang zijn.
- **Datumprikker**; dit staat voor de reeds bestaande applicatie Datumprikker, met de bijbehorende webpagina's.
- **SmA's**; dit staat voor alle SmA's waarmee gecommuniceerd moet worden.
- **Implementation**; hierbinnen zitten alle implementaties, oftewel de plug-ins.



Figuur 3, initieel ontwerp

2 Uitvoering

Het bouwen van de plug-in module en de bijbehorende implementaties is in 5 sprints uitgevoerd. Bij elke sprint is er van te voren gekeken wat er tijdens deze sprint gedaan zou worden. Deze keuze werd bepaald door de storymap. De uiteindelijke versie hiervan is te zien in **Figuur 4**. De eisen die in de storymap voorkomen, komen uit het document **Requirements**. De meeste hiervan waren al opgesteld voordat er met de uitvoering begonnen werd, maar sommige requirements kwamen tijdens het ontwerpen en bouwen naar voren.

F1a		F1b	F1d	Nb1	F2e
Er moet ingelogd kunnen worden met behulp van de SmA's.		Er moeten contacten van de gebruiker opgehaald kunnen worden met behulp van de SmA's.	Er moeten berichten naar bekende accounts van de gebruiker gestuurd kunnen worden met behulp van de SmA's.	De plug-in module moet volgens de use cases handelen.	Er moet een afspraak toegevoegd kunnen worden aan de kalender van de SmA's.
100%		100%	100%	100%	100%
F2	F3	F1c	No4	No5	No6
Er moet een module gebouwd worden die plugins kan bevatten.	De plugins dienen elk met 1 SmA te kunnen communiceren.	Er moet agenda data van de gebruiker opgehaald kunnen worden met behulp van de SmA's.	De website(s) van Datumprikker moeten weten wat voor type object (van No2) er ontvangen zal worden.	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om aan de data van de SmA's te komen.	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om data te versturen naar de SmA's.
F4	F5a	No2			
De plug-ins moeten JSON accepteren van de SmA's.	De plug-ins moeten met REST om kunnen gaan.	De requirements F2a, F2b, F2c, F2d, F2e moeten los van elkaar geïmplementeerd kunnen worden.			
F5c	F6	No3			
De plug-ins dienen met Oauth V2 om kunnen gaan.	De van de SmA's ontvangen data moet omgezet worden naar een generiek object.	Het generieke object van No2 zal per implementatie moeten verschillen.			
No1	Nv5				
De module moet een opzichzelfstaande structuur hebben.	De authenticatie dient gedaan te worden met Oauth V2				
F7	F8				
Templates moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.	Plugins moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.				
Nv1	Nv2				
De module moet gekoppeld zijn aan een bestaand developer account van de social media.	De module moet geregistreerd zijn bij de social media.				
Nv3					
De module moet in de developers console geregistreerd worden binnen een bestaand account van de social media.					

Figuur 4, uiteindelijke storymap[4]

[4] Volledige versie is te zien in bijlage 1

In de hierop volgende 5 paragrafen worden de sprints besproken. Per sprint zullen de volgende dingen naar voren komen:

- De periode waarin de sprint plaatsvond.
- De requirements waaraan de sprint uiteindelijk moest voldoen.
- De problemen en/of de daarbij horende keuzes/oplossingen.
- De wijzigingen in het ontwerp.

2.1 Sprint 1: Inloggen

Periode:

Sprint 1	Inloggen														
week	3					4					5				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Requirements uit storymap:

Code	Requirement
F1a	Er moet ingelogd kunnen worden met behulp van de SmA's.
F2	Er moet een module gebouwd worden die plug-ins kan bevatten.
F3	De plug-ins dienen elk met 1 SmA te kunnen communiceren.
F4	De plug-ins moeten JSON accepteren van de SmA's.
F5a	De plug-ins moeten met REST om kunnen gaan.
F5c	De plug-ins dienen met Oauth V2 om kunnen gaan.
F6	De van de SmA's ontvangen data moet omgezet worden naar een generiek object.
No1	De module moet een opzichzelfstaande structuur hebben.
Nv5	De authenticatie dient gedaan te worden met Oauth V2
F7	Templates moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.
F8	Plug-ins moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.
Nv1	De module moet gekoppeld zijn aan een bestaand developer account van de social media.
Nv2	De module moet geregistreerd zijn bij de social media.
Nv3	De module moet in de developers console geregistreerd worden binnen een bestaand account van de social media.

De focus bij deze sprint lag bij het mogelijk maken om te kunnen inloggen bij een SmP. Indien een gebruiker gebruik wil maken van deze functionaliteit, dan dient deze de volgende stappen te ondernemen:

1. Een keuze maken uit de beschikbare SmA's.
2. De knop van de bijbehorende SmA aanklikken.
3. De inloggegevens in het opkomende scherm invullen.
4. De rechten accepteren die de module nodig heeft voor het gebruik van de SmA.

Om deze stappen mogelijk te maken moesten er de volgende onderdelen gebouwd worden:

- Een (test)webpagina die de knoppen en de schermen toont en afhandelt;
- Een correct ingestelde 'app'[5] bij de mogelijke SmP's;
- Een plug-in module met de volgende onderdelen:
    - Implementaties voor het inloggen;
    - Een manier om te communiceren, authentiseren en converteren (van de ontvangen data) met de SmA's;

Er moest dus al een heel groot gedeelte van de plug-in module functioneel zijn om aan de requirements van deze sprint te kunnen voldoen. Door vooral veel gebruik te maken van overerving kon de hoeveelheid werk echter redelijk beperkt blijven. Daarnaast zorgde dit er voor dat er in de toekomst gemakkelijk doorgebouwd kon worden, zonder al te veel aanpassingen te maken binnen de onderdelen die in deze sprint gemaakt werden.

2.1.1 Requirements

Voordat de sprint begon, zijn er een aantal nieuwe requirements opgesteld:

id	Requirement
F2	Er moet een module gebouwd worden die plug-ins kan bevatten.
F3	De plug-ins dienen elk met 1 SmA te kunnen communiceren.
F6	De van de SmA's ontvangen data moet omgezet worden naar een generiek object.
No1	De module moet een opzichzelfstaande structuur hebben.
F7	Templates moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.
F8	Plug-ins moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.

2.1.2 Keuzes en problemen

Tijdens het bouwen bleek het ontwerp niet te voldoen. Dit kwam doordat er een aantal dingen anders bleken dan eerst gedacht. Zo bleek onder andere het authentiseren met OAuth2 qua programmeren praktisch geen (extra) werk te kosten indien er met REST gewerkt wordt. Hierdoor kon het eigenlijk niet anders dan deze 2 onderdelen samen te voegen.

Indien er in de toekomst een andere manier van communiceren in de plug-in module geïmplementeerd wordt, zal er gekeken moeten worden of het authentiseren wel los(ser) gebouwd kan worden. Bij een nieuwe vorm van authentiseren zal dit sowieso bekeken moeten worden.

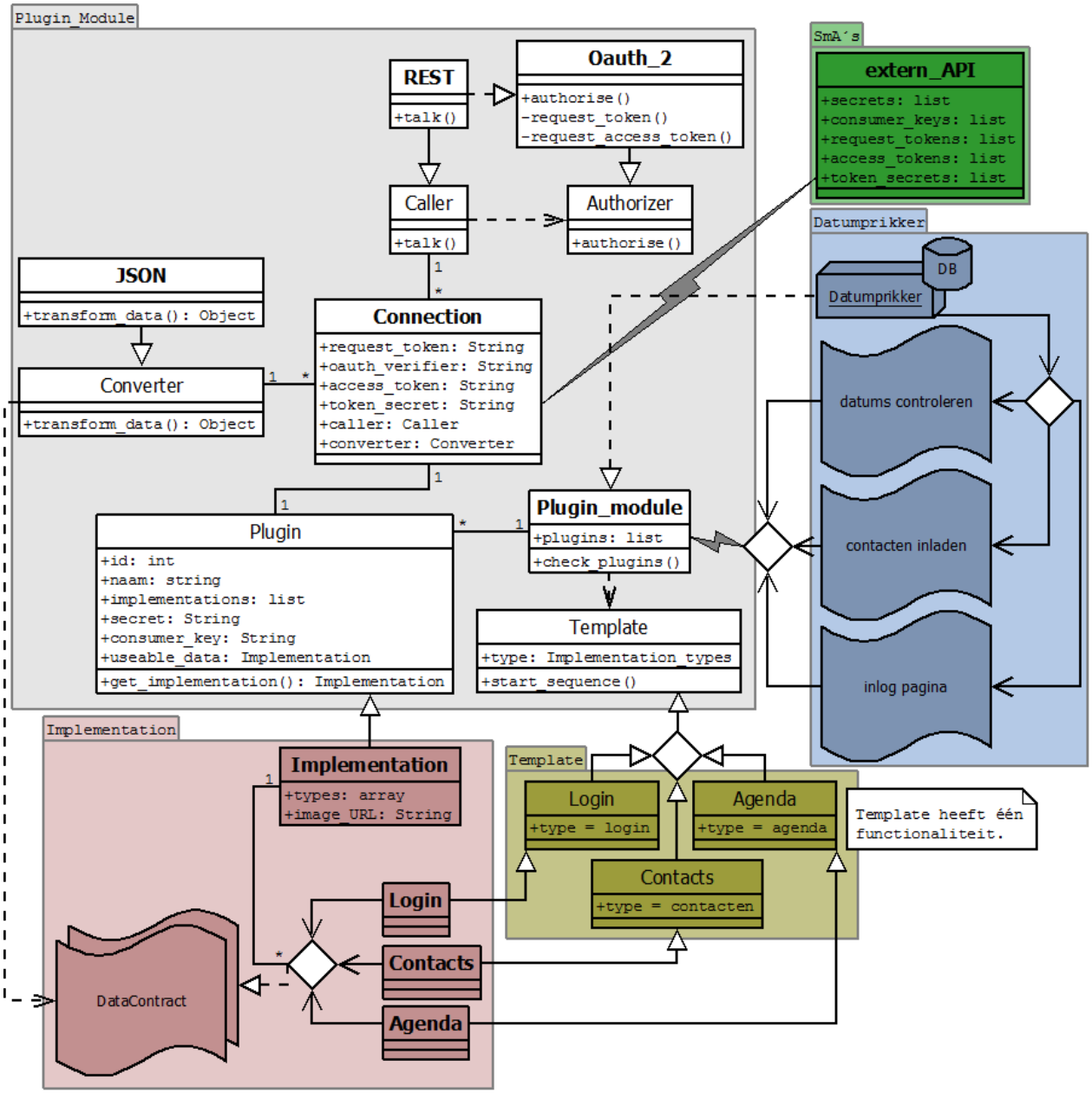
Een ander probleem was dat het converteer onderdeel anders in elkaar zat dan eerst gedacht. Om de ontvangen data te kunnen gebruiken, moet dit namelijk omgezet worden naar specifieke klassen. Aan het begin van het project was er vanuit gegaan dat alle data (gemakkelijk) omgezet kon worden naar één generiek object. Dit laatste is nog steeds mogelijk, maar er komt nog wel één stap tussen: de ontvangen data via een data contract omzetten naar een (specifieke) klasse.

[5] Dit is niets meer dan een (developer) account bij de betreffende SmP. Dit account bevat de gegevens die nodig zijn voor het gebruik van de SmA's

Ontwerp

Deze 2 punten zorgde voor een behoorlijke wijziging in het ontwerp, zoals te zien in **Figuur 5**. Hierin is te zien dat er aan de **Implementation** package data contracten zijn toegevoegd. Deze worden gebruikt door de **Converter** klasse. Daarnaast heeft de **Caller** klasse nu een dependency bij de **Authorizer** klasse.

Een andere grote wijziging in het ontwerp is de **Template** package. Deze is toegevoegd om er voor te zorgen dat de implementaties overerven van generieke templates. In deze templates wordt de functionaliteit bepaald, zoals bijvoorbeeld het inloggen.



Figuur 5, ontwerp versie 0.3

SmA's

Tijdens het bouwen waren er 2 SmA's die voor complicaties zorgden. Dit kwam voornamelijk door de documentatie die op de SmP-websites stond. Om te kunnen weten wat er verstuurd wordt door de SmA, moet de documentatie aan de SmA kant namelijk volledig en correct zijn. Dit was niet het geval bij Microsoft. De documentatie was zeer beperkt, waardoor er veel meer werk voor nodig was om dit te implementeren dan nodig zou zijn.

Daar kwam ook nog eens bij dat Microsoft geen localhost ondersteunt als callback-URL voor toegang met de SmA. Hierdoor werd het nog lastiger om te achterhalen wat Microsoft verwacht en verstuurt, omdat er niet op code niveau gedebugged kan worden. Uiteindelijk is er doormiddel van cookies achterhaald wat Microsoft exact terugstuurt. Voordat dit resultaat er echter uitrolde, moest eerst het aanspreken van de SmA zelf opgelost worden.

In tegenstelling tot andere SmA's is Microsoft hier namelijk zeer streng (en des te onduidelijker) over. En indien er niet aan deze (ongeschreven) regels voldaan wordt, geeft de SmA een generieke HTTP 415[6] foutmelding. Met deze foutmelding wordt alleen aangegeven dat er iets binnen de request niet voldoet aan de regels, maar wat de precieze afwijking van die regels is, staat niet vermeld bij die melding. Na op verschillende websites gezocht te hebben naar een oplossing, werd deze gevonden dankzij een soortgelijk probleem van iemand anders[7]. Door de aanroep van die persoon te vergelijken met de zelfgemaakte aanroep, bleek dat er een variabele verkeerd ingesteld werd.

Facebook

Bij Facebook was er een ander probleem. Hoewel deze zich volledig houdt aan de OAuth2 regels (en zelfs mee geholpen heeft met het opstellen hiervan), zit er een inconsistentie bij het authentiseren. Hierbij wordt er namelijk geen JSON geretourneerd, maar een stuk tekst in URL query formaat. Hierdoor ontstond er tijdens het converteren een foutmelding, aangezien er JSON verwacht werd.

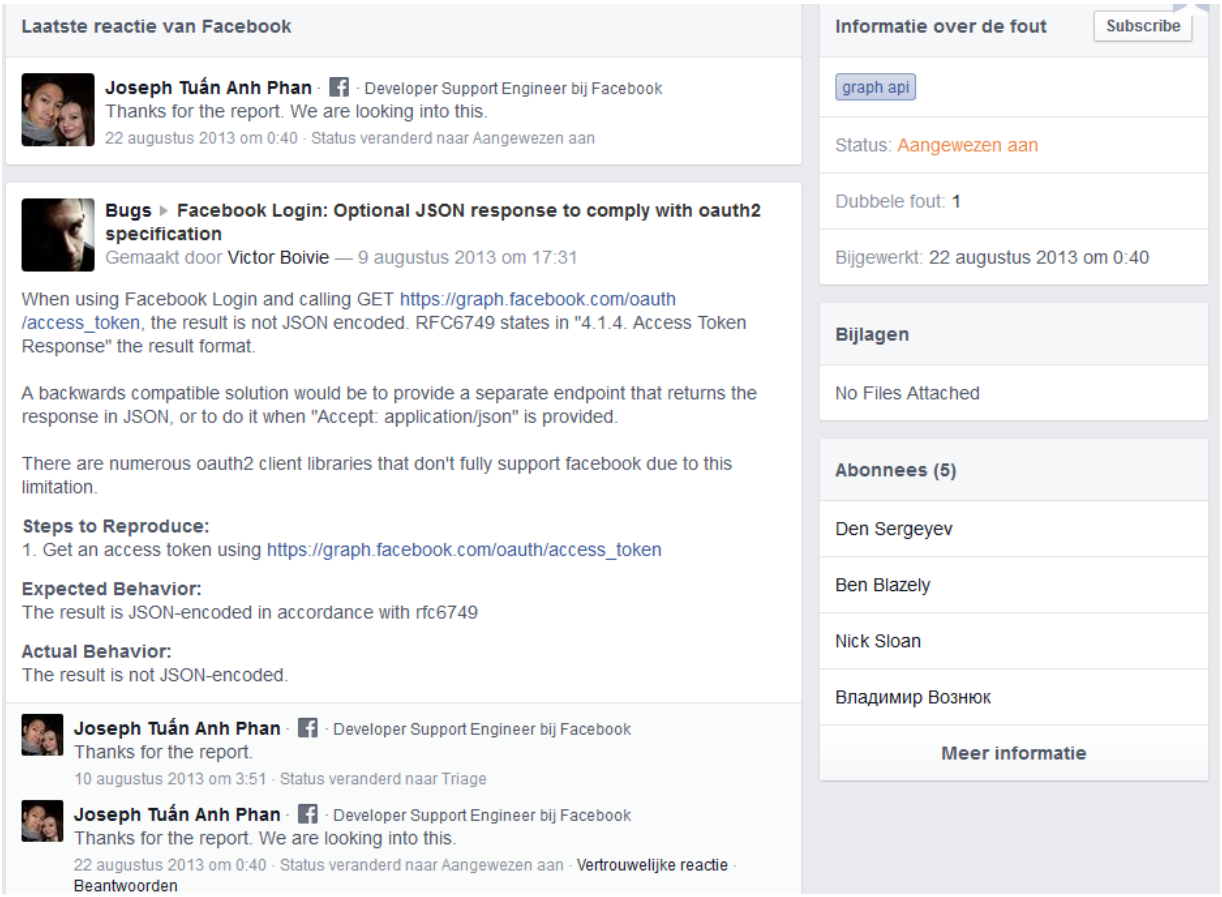
Nadat de geretourneerde data bekeken werd en er geconcludeerd kon worden dat het niet aan de gemaakte code lag, is er op het internet gezocht naar mogelijke oorzaken/oplossingen. Al snel bleek dat het probleem bij Facebook lag. De betreffende bug is al een jaar lang bekend bij Facebook (zie **Figuur 6**), maar omdat de SmA al live is, zijn aanpassingen zeer lastig en duur (alle (externe) partijen moeten de code weer aanpassen).

Een (simpele) oplossing voor dit probleem werd niet gegeven, dus er moest zelf iets bedacht worden. Uiteindelijk is de volgende constructie gemaakt:

- De (van Facebook) ontvangen data wordt gecontroleerd op de regels van JSON (zoals bijvoorbeeld het gebruik van '{' en '}').
- Indien er niet aan deze regels voldaan wordt:
  - Pas de data zo aan zodat het aan de regels van JSON voldoet.
- Converteer de JSON naar de generieke klasse doormiddel van de data contracten.

[6] Media-type niet ondersteund (http://www.w3.org/Protocols/rfc2616/rfc2616-sec10.html)
[7] http://social.msdn.microsoft.com/Forums/onedrive/en-US/39e8e288-4775-4e9f-a957-3fe7ba05b7d3/problem-while-trying-to-add-a-new-contact-via-rest-api?forum=messengerconnect

Hoewel het omslachtig is om de data twee keer om te zetten naar iets anders, betekent dit wel dat alle data die van de Facebook SmA binnenkomt hiermee opgevangen kan worden. Daarnaast kunnen er geen problemen ontstaan indien Facebook de bug (ooit) oplost.



Figuur 6, Facebook authentiseer bug [8]

[8] Bron: https://developers.facebook.com/x/bugs/162050973983689/ (voor deze link moet er ingelogd worden bij Facebook)

2.2 Sprint 2: Contacten en kalender ophalen

Periode:

Sprint 2	Contacten en Kalender ophalen														
week	5					6					7				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Storymap:

Code	Requirement
F1b	Er moeten contacten van de gebruiker opgehaald kunnen worden met behulp van de SmA's.
F1c	Er moet agenda data van de gebruiker opgehaald kunnen worden met behulp van de SmA's.
No2	De requirements F2a, F2b, F2c, F2d, F2e moeten los van elkaar geïmplementeerd kunnen worden.
No3	Het generieke object van No2 zal per implementatie moeten verschillen.

De focus van deze sprint lag bij het bouwen van 2 functionaliteiten:

- Contacten ophalen van de gebruiker (van de SmA's)
- Kalender data ophalen van de gebruiker (van de SmA's)

De reden dat er 2 functionaliteiten gebouwd werden, is omdat het hier alleen om de implementaties gaat. Er werd niet verwacht dat er (grote) aanpassingen nodig waren om deze in te bouwen. Wel zouden er aanpassingen gemaakt moeten worden aan de testwebpagina, zodat de nieuwe functionaliteiten getest en gebruikt konden worden.

2.2.1 Requirements

Voordat de sprint begon, zijn er een aantal extra requirements opgesteld:

id	Requirements
No2	De requirements F2a, F2b, F2c, F2d, F2e moeten los van elkaar geïmplementeerd kunnen worden.
No3	Het generieke object van No2 zal per implementatie moeten verschillen.

2.2.2 Keuzes en problemen

In tegenstelling tot het inloggen en ophalen van contacten, moet er bij het ophalen van kalender data (eerst) data gestuurd worden naar de SmA. De data die gestuurd wordt, zijn de begin- en einddatum met de bijbehorende tijden.

Het idee hierachter is dat de gebruiker een aantal datum mogelijkheden aangeeft voor een afspraak, en dat daarna gecontroleerd wordt of er al afspraken staan op die momenten. Om te voorkomen dat er continu data opgevraagd wordt aan de SmA, wordt er, op basis van de aangegeven datums, een minimum en maximum opgezocht. Daarna wordt er bij de kalenders gefilterd met deze twee datums.

Op de webpagina moest er daarom naast de nieuwe knoppen een mogelijkheid komen om datums in te vullen. Tevens moest er iets gebouwd worden wat kon aangeven of er afspraken (van de SmP's) waren die binnen de range van die datums vielen.

Filter

Het is niet handig om alle datums van een gebruiker op te vragen aan een SmA. Er werd daarom besloten om te bepalen wat de range is waarop gezocht moet worden. Deze range komt tot stand door te controleren wat de minimum en maximum datums zijn die een gebruiker invult. Deze 2 datums kunnen dan gebruikt worden om te filteren op de kalender van de SmA.

Om (generiek) te kunnen filteren, moest er een oplossing gevonden worden om gemakkelijk filters toe te voegen. Daarom is er (ook in het ontwerp) een filter-klasse toegevoegd. Door deze klasse toe te voegen werd het gelijk ook mogelijk gemaakt om (in de toekomst) binnen andere implementaties te filteren op data.

Facebook

Bij de Facebook Graph-API (de SmA die gebruikt werd) is het onmogelijk om bij evenementen (de Facebook variant op een kalender) te filteren op de data. Er is wel een filter mogelijkheid bij het zoeken naar evenementen, maar alleen als je zoekt op alle evenementen van iedereen. Als er alleen naar de evenementen van de gebruiker gezocht moet worden, dan kan dit niet. Waarom dit het geval is, is onduidelijk. Onderzoek op het web leverede alleen 'oplossingen', maar geen redenen.

De oplossing die over het algemeen genoemd werd, was om te filteren nadat de data ontvangen is. Dit houdt dus in dat Facebook eerst alle evenementen van de gebruiker retourneert, en dat er dan binnen eigen code gefilterd wordt op deze data. Dit is zeer omslachtig omdat er dan iedere keer onnodige data verstuurd wordt, en dat er daarna ook nog eens zelf een filter functie gebouwd moet worden.

Hoewel het versturen van data geen invloed leek te hebben op de performance[9], en het bouwen van de filter functie ook niet zeer ingewikkeld was, is het natuurlijk slordig van Facebook om dit probleem bij de SmA-aanroeper te leggen. Zeker omdat er bij andere delen wel filter mogelijkheden zijn.

Datum/tijd (ISO 8601 [10])

Een ander probleem waar tegenaan gelopen werd tijdens deze sprint, was het afhandelen van datum/tijd variabelen. Dit kwam door twee aspecten. Allereerst was er het probleem van conversie tussen Javascript (front-end) en C# (back-end): Javascript houdt namelijk de UNIX-tijd aan, terwijl C# de volledige ISO 8601 standaard aanhoudt. Hierdoor moest er continu rekening gehouden worden met het verschil.

Het andere aspect had met Datumprikker zelf te maken. Datumprikker heeft namelijk (tot nu toe) datum/tijd altijd opgeslagen in de database, zonder het vermelden van de tijdzone. Dit maakte voorheen nooit veel uit, aangezien de gebruikers van Datumprikker (nagenoeg) allemaal binnen dezelfde tijdzone (UTC +01:00 of UTC +02:00) afspraken plannen.

---

[9] Het ophalen van data ging even snel als bij de andere SmA's. Maar het ging hierbij wel steeds om slechts één request. Het is daarom niet met zekerheid te zeggen dat dit ook bij grote (simultane) groepen requests ook het geval is.

[10] Bron: https://web.archive.org/web/20110614235056/http://www.iso.org/iso/support/faqs/faqs_widely_used_standards/widely_used_standards_other/date_and_time_format.htm

Het gebruik van social media verandert dit echter. Dit komt doordat de SmA's data op verschillende locaties op kan slaan. Hierdoor verwachten zij dan ook dat de tijdzone aangegeven wordt. Om converteerproblemen te voorkomen, is er voor gekozen om alle datums om te zetten naar UTC. Hierdoor is er meer consistentie, en daarnaast scheelt het bewerking. Uiteindelijk wordt alle data namelijk in string-vorm verstuurd. Dit houdt in dat bij een (van UTC) afwijkende tijdzone er meer getallen moeten bijkomen. Indien het UTC betreft, hoeft er echter alleen maar een 'Z' achter de datum/tijd geplaatst te worden. Voorbeelden hiervan zijn:

UTC	Nederlandse notatie	ISO 8601
+01:00	22 februari 2014, 09:00:00	2014-02-22T09:00:00+01:00
+01:00	1 mei 2014, 10:23:56	2014-05-01T09:23:56Z
-05:00	14 juli 2014, 01:30:00	2014-07-13T20:30:00Z
+02:30	31 december 2013, 22:00:00	2014-01-01T00:30:00Z

Offline token

Eén van de grootste voordelen van het gebruik van SmA's, is (logischerwijs) dat er gebruik gemaakt kan worden van externe data. De grote vraag is echter: wil je ook gebruik maken van deze data indien de gebruiker niet (meer) is ingelogd bij de SmA?

Deze vraag kwam niet naar voren bij de vorige sprint, omdat de ontvangen data bij het inloggen sowieso bij Datumprikker zelf opgeslagen moet worden om een koppeling te kunnen maken met (bestaande) Datumprikker accounts.

Voor het ophalen van de (contact en kalender) data speelt deze vraag echter wel een belangrijke rol, vooral als er vooruit gekeken wordt naar de hierna volgende sprints. Als er bijvoorbeeld (automatische) berichten gestuurd gaan worden naar de contacten, of als er (automatisch) afspraken in de kalender geplaatst/gewijzigd worden, dan zal er toegang moeten zijn tot deze data. Deze (behalve natuurlijk de essentiële) data zal zelf niet opgeslagen worden bij Datumprikker om 2 redenen:

1. Overbodige opslag/database communicatie; aangezien de data al opgeslagen is bij de SmA, hoeft dit niet opgeslagen te worden bij Datumprikker. Als al deze data wel opgeslagen zou worden, dan zou dit betekenen dat er veel meer ruimte gereserveerd/gebruikt moet gaan worden. Daarnaast moet er dan iedere keer gecommuniceerd worden met de database, die nu toch al zwaar bevraagt wordt door het 'normale' gebruik.
2. Nieuwe data moet toch naar de SmA gestuurd worden; als er bij de SmA nieuwe data toegevoegd moet worden (kalender afspraak/ bericht), dan zal er met de SmA gecommuniceerd moeten worden. Daarvoor is er uiteraard toegang nodig.

Deze laatste reden is ook waar de hele discussie door begon. De opdrachtgever wilde het versturen van berichten, en het inplannen van afspraken, namelijk automatisch laten verlopen. Dit hield in dat de (hoofd)gebruiker van de afspraak geen handeling hoefde te doen, en dus niet online hoefde te zijn.

Dit betekent dat de (tijdelijke) toegangstoken voor het gebruik van de gegevens die standaard door de SmA's gegeven wordt niet toereikend genoeg zou zijn. De opdrachtgever dacht echter dat de token voor altijd geldig zou zijn. Na wat research bleek dat dit vroeger (bij andere methodes) wel het geval te zijn, maar sinds OAuth2 door alle SmA's gebruikt wordt, is dit niet meer het geval.

Om toch langer toegang te hebben tot de data houden de SmA's verschillende manieren aan:

- Google: indien een extra recht meegestuurd wordt, wordt er een refresh-token mee teruggestuurd. Met deze refresh-token kunnen dan weer nieuwe (refresh-)tokens aangevraagd worden. Hierdoor kan er dus zonder tussenkomst van de gebruiker gebruik gemaakt worden van de data.
- Facebook & Microsoft: door bij het authentiseren de oude token mee te sturen, wordt er een nieuwe token gestuurd waarmee er toegang verleend wordt.
- LinkedIn: de enige manier om de token bij LinkedIn te vernieuwen werkt niet voor OAuth2. De token is daarentegen veel langer geldig dan bij de andere SmA's. Deze is namelijk 60 dagen geldig. Indien er na deze 60 dagen toch nog gebruik gemaakt moet worden van de data, dan moet de gebruiker opnieuw inloggen.

LinkedIn

LinkedIn maakt dus geen gebruik van het 'refresh'-idee. Hierdoor werd het offline-access concept in ieder geval voorlopig in de ijskast gezet. Mocht er later dan toch gebruik gemaakt worden van de offline-access (en LinkedIn dit nog steeds niet ondersteunt), dan zal er waarschijnlijk bij het verlopen van de token een e-mail gestuurd worden naar de (hoofd) gebruiker.

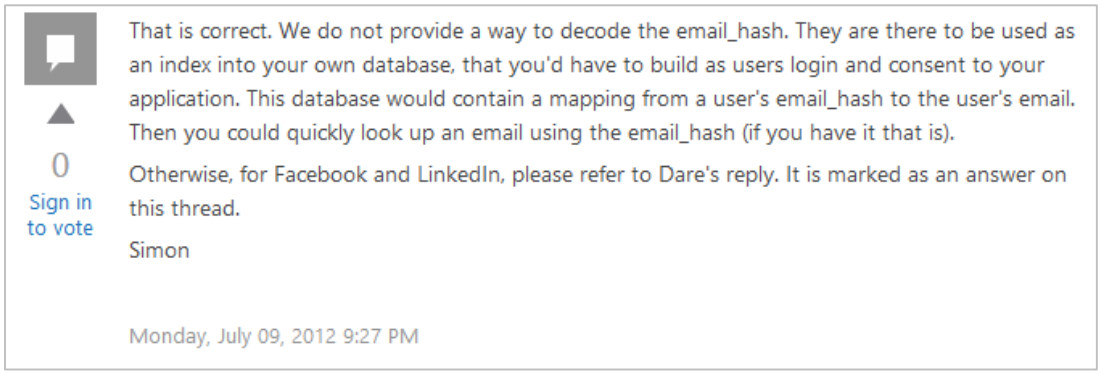
Een ander 'probleem' bij LinkedIn is dat ze geen email adressen van contacten sturen. In plaats daarvan moeten alle berichten via de SmA gaan. Hiervoor zijn de account-id's nodig van de contacten. Dit is (voor nu) dus geen probleem, maar zal bij de volgende sprint aangepakt moeten worden.

Als laatste heeft LinkedIn geen kalender functie, dus hiervoor hoefde dan ook geen kalender functionaliteit gebouwd te worden.

Microsoft

Microsoft gaf op een ander vlak problemen. Indien er om de contacten gevraagd werd, werden de emailadressen van die contacten niet meegestuurd. In plaats daarvan werden er hashes gestuurd. Het idee wat hier achter zit is ergens wel begrijpelijk, maar vooral zeer omslachtig:

'De hashes zijn bedoeld om op te slaan in de (eigen) database. Indien een hash overeenkomt, dan betreft het dus dezelfde contactpersoon.' [11]



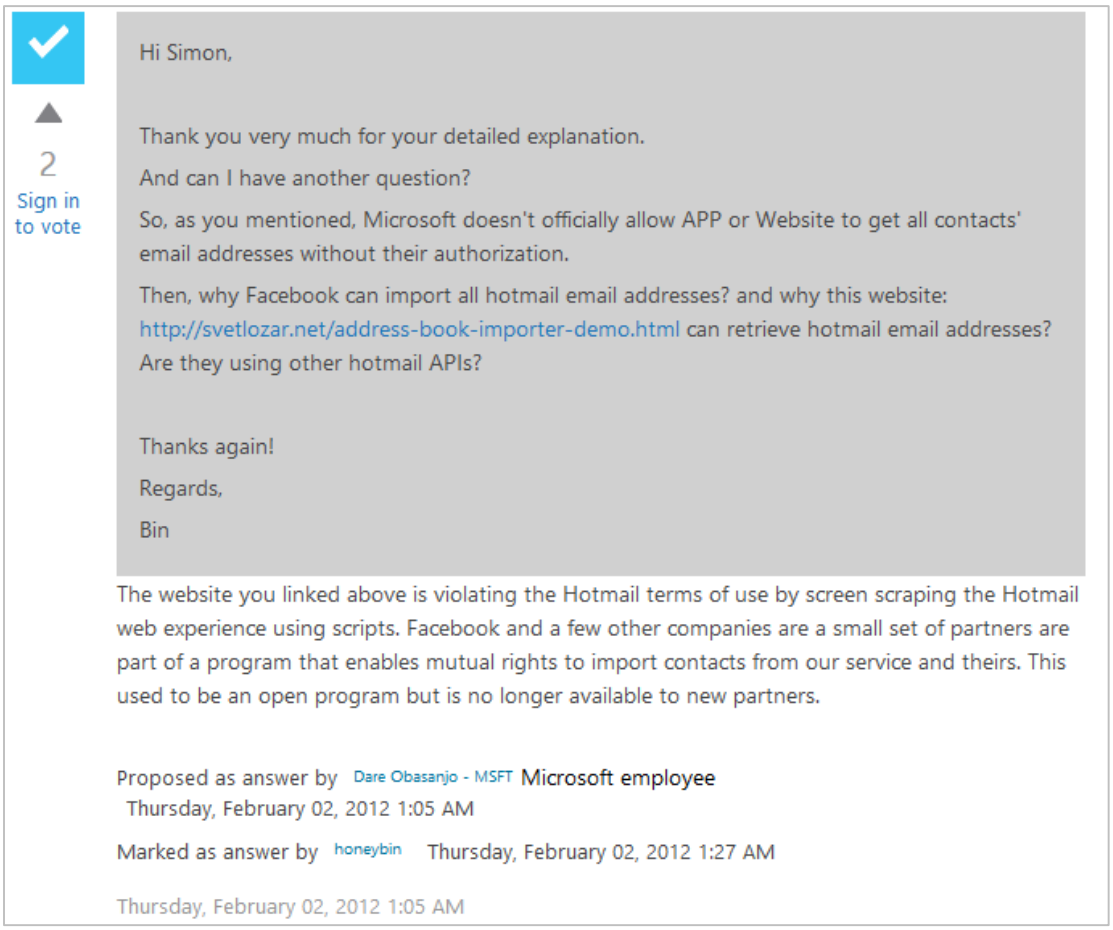
Figuur 7, forum antwoord op soortgelijke vraag [11]

[11] Bron: http://social.msdn.microsoft.com/Forums/live/en-US/c6dcb9ab-aed4-400a-99fb-5650c393a95d/how-retrieve-users-contacts-email-address

In andere woorden: je kunt alleen een email adres achterhalen indien deze al bekend is binnen je eigen systeem/database. Dit is een veilig concept, ware het niet dat het hele idee van het gebruik van de SmA is om contact gegevens op te halen en te kunnen gebruiken.

In theorie zouden de contact gegevens natuurlijk via andere wegen verkregen kunnen worden. Sterker nog: er zijn websites die contacten, inclusief het email adres, kunnen inladen. Dus het zou gewoon moeten kunnen. Hierover zegt Microsoft echter het volgende:

'Er is een zeer beperkte groep van partners die toegang hebben tot de email adressen. Vroeger was er open toegang, maar deze toegang geldt niet meer voor nieuwe partners.' [11]



Figuur 8, ander bericht in hetzelfde forum [11]

Zoals te zien in **Figuur 7** en **Figuur 8** waren deze berichten van 2012. Het is onduidelijk of dit beleid veranderd is, of dat Microsoft zichzelf tegenspreekt. De email adressen kunnen namelijk gewoon opgevraagd kunnen worden… Mits de juiste variabelen meegestuurd worden. Er moet namelijk een extra recht aangevraagd worden om toegang te krijgen. Dit staat echter slecht gedocumenteerd, en is (na lang zoeken) alleen gevonden op een externe site[12].

[12] http://stackoverflow.com/questions/7370141/how-to-convert-hash-into-text-from-the-response-of-live-api-for-contacts

Email-adressen Facebook

Bij Facebook was er ook wat aan de hand rondom de email-adressen. Dit keer zorgde Facebook echter niet zelf voor complicaties, maar lag het probleem bij de aannames. Eén van de stakeholders dacht namelijk dat het niet (meer) mogelijk was om met Facebook e-mails te sturen. Deze aanname kwam door een artikel op Tweakers[13]. In het artikel gaat het echter over de eigen emaildienst van Facebook, niet over het verzenden van (e-mail) berichten. Oftewel: het emailadres persoon@facebook.com bestaat niet meer. Maar als "persoon" een bestaand accountnaam is, dan kan er hier wel een bericht naar gestuurd worden. Dit wordt in eerste instantie op de pagina van de persoon gezet, tenzij deze niet online is. Dan wordt het bericht doorgestuurd naar het bij Facebook bekende emailadres van die persoon (bijvoorbeeld persoon@gmail.com).

Documentatie SmA's

Als er één conclusie getrokken moest worden over deze sprint, dan was het wel:

"groot bedrijf" !== "grote bron van informatie"

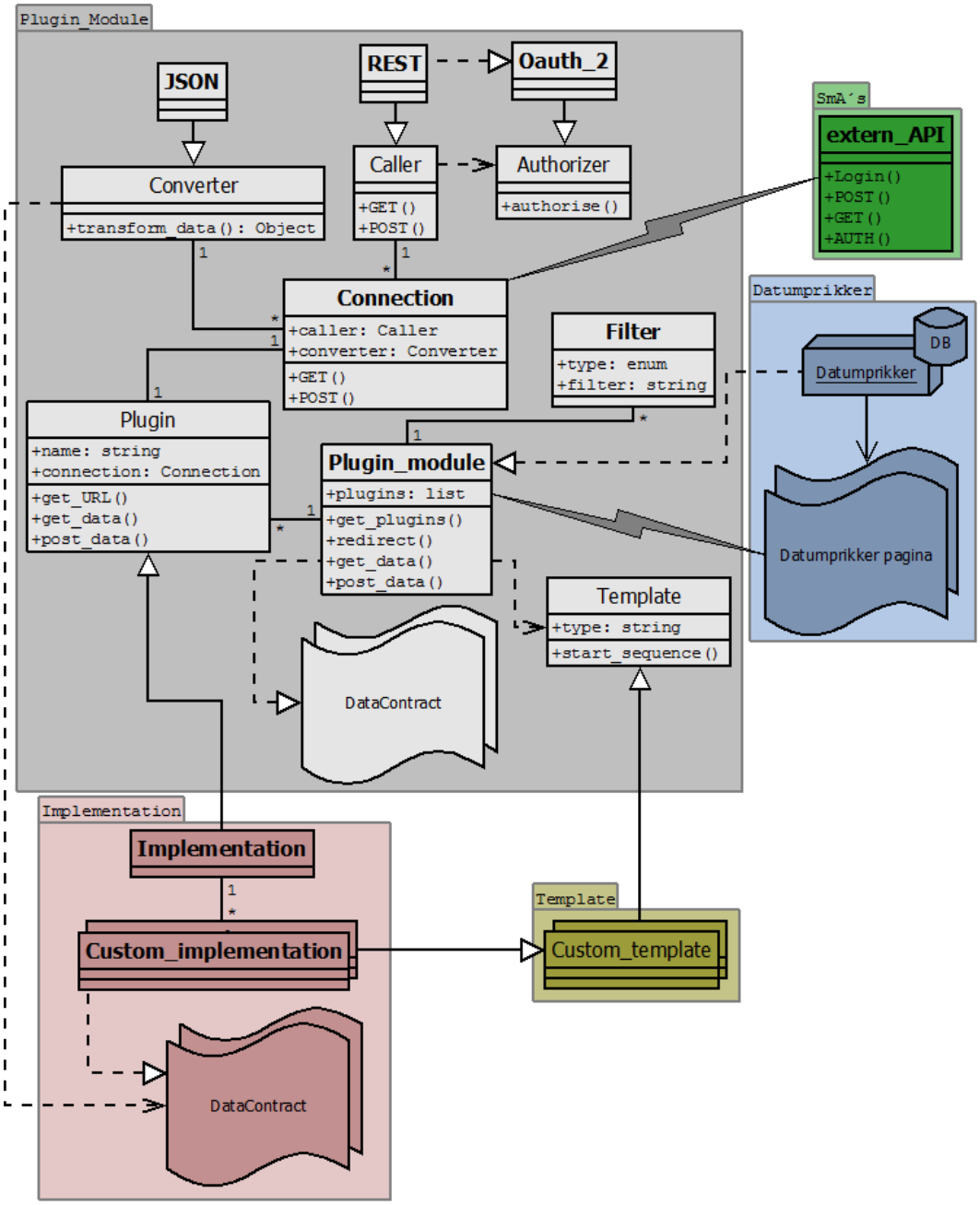
Alle vier de bedrijven (Google, Facebook, Microsoft en LinkedIn) hebben simpel gezegd slechte documentatie. De enige reden waarom Google tot nu toe niet besproken is, komt doordat, hoewel de documentatie hier en daar wat gebrekkig is, er in ieder geval geen fouten in staan. Daarnaast heeft Google bij de meeste API's een live demo waarmee direct de in- en uitvoer te zien is. De andere SmA's hebben ook zo'n demo applicatie op hun website, maar deze werkt of gebrekkig, dan wel de in-/uitvoer is onduidelijk.

[13] Artikel: http://tweakers.net/nieuws/94529/facebook-sluit-zijn-e-maildienst.html

Ontwerp

Het ontwerp is tijdens deze sprint enigszins aangepast. Een aantal belangrijke wijzigingen zijn:

- De **Filter** klasse is toegevoegd. Hiermee kan er gefilterd worden op specifieke data.
- De templates en implementaties zijn nu zo vorm gegeven dat het duidelijker is hoe deze werken.
- De **Datumprikker pagina**'s zijn samengevoegd om meer eenvoud te creëren.
- Er is een **DataContract** voor de **Plugin_module** bijgekomen. Dit is gedaan om er voor te kunnen zorgen dat de **Datumprikker pagina**'s een generiek object kunnen verwachten. Dit zal bij de volgende sprint exact(er) uitgewerkt worden.



Figuur 9, ontwerp versie 1

2.3 Sprint 3: Berichten verzenden

Periode:

Sprint 3	Berichten verzenden														
week	6					7					8				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Storymap:

Code	Requirement
F1d	Er moeten berichten naar bekende accounts van de gebruiker gestuurd kunnen worden met behulp van de SmA's.
No4	De website(s) van Datumprikker moeten weten wat voor type object (van No2) er ontvangen zal worden.

De focus van deze sprint lag bij de functionaliteit om berichten te kunnen versturen. Deze functionaliteit zou de laatste verplichte zijn die gebouwd zou worden. Na deze sprint zou het proof-of-concept dus af moeten zijn. Hoewel het verzenden van berichten niet bepaald spannend lijkt, komt er toch meer bij kijken. Zo is dit de eerste functionaliteit waarbij een gebruiker meerdere SmA's 'tegelijk'[14] moet kunnen gebruiken. Daarnaast wordt er bij deze functionaliteit daadwerkelijk iets gepost, in tegenstelling tot bijvoorbeeld het ophalen van kalender-data, waarbij er alleen een filter etc. verstuurd hoeft te worden.

2.3.1 Requirements

Voordat deze sprint begon, is er één nieuwe requirement bijgekomen:

id	Requirements
No4	De website(s) van Datumprikker moeten weten wat voor type object (van No2) er ontvangen zal worden.

2.3.2 Keuzes en problemen

Dit alles zorgde voor verschillende uitdagingen, zoals bijvoorbeeld bij de LinkedIn SmA. Dit is namelijk de enige SmA die geen e-mails geeft van contacten. Het is daarom ook de enige SmA waarbij er daadwerkelijk een implementatie moest komen voor berichten versturen. Hoewel berichten versturen oorspronkelijk onder de contacten-implementatie zou vallen, is er gekozen om hiervoor toch een aparte implementatie te maken. De reden hiervoor is dat het totaal andere functionaliteit betreft, met andere/extra eisen.

Om berichten te kunnen versturen, moesten er 4 veranderingen doorgevoerd worden:

- **POST**
- **Data contract**
- **Webpagina**
- **Back-end**

[14] Een gebruiker kan meerdere SmP's selecteren voor een bepaalde actie. De SmA's hoeven niet exact tegelijk (multi-threaded) aangeroepen te worden.

Daarnaast was het versturen van data van de back-end naar de front-end niet op orde. Er werd hierbij namelijk ook gevoelige informatie gestuurd die niet voor de front-end bedoeld is. Dit was handig in het begin om te kunnen controleren of alles doorkwam, maar gezien het feit dat de module bijna klaar was, was deze informatie niet meer gewenst. Om dit op te lossen, werden er Interfaces en een **Database object** aangemaakt.

Uiteraard betekende al deze veranderingen dat ook het **Ontwerp** weer aangepast moest worden.

POST

Het POST gedeelte van de plug-in module moest flink aangepast worden om de vele variabelen die hierbij horen te kunnen verwerken. In tegenstelling tot de kalender functionaliteit moesten er nu namelijk de volgende dingen (extra) meegegeven worden:

- Contact(en)
- Onderwerp
- Bericht (met al dan niet een URL erbij)

Daarnaast moest er rekening gehouden worden met toekomstige implementaties, dus het simpelweg toevoegen van deze variabelen aan de methode-aanroep was geen optie.

Data contract

Daarom werd er voortgeborduurd op het aangepaste ontwerp van de vorige sprint, en werden er data contracten aangemaakt. Deze data contracten konden ingevuld worden door de front-end, en meegestuurd worden naar de back-end. Hierbinnen zouden de variabelen dan gebruikt kunnen worden om de berichten te kunnen versturen. Met het gebruik van de data contracten konden er gemakkelijk nieuwe variabelen toegevoegd worden, zonder de interne code te moeten aan te passen.

Webpagina

Aangezien de front-end de datacontracten moest invullen, betekende dit dat deze hiervoor geschikt gemaakt moest worden. Dit hield in dat de volgende onderdelen toegevoegd moesten worden:

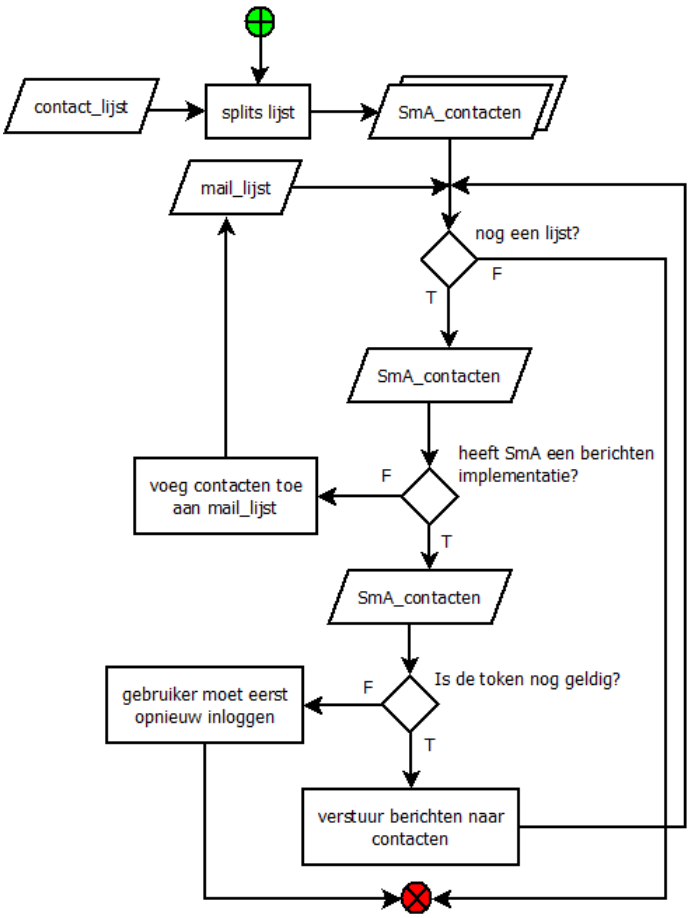
- Invoervelden voor een bericht en titel; hierin moest het bericht opgesteld kunnen worden dat verstuurd moest gaan worden. Besloten werd om dit in een apart scherm te zetten.
- Mogelijkheid om contacten te selecteren; de contacten die opgehaald werden met behulp van de functionaliteit van de vorige sprint, moesten geselecteerd kunnen worden. Aangezien een gebruiker van verschillende SmA's contacten op kan halen, moeten de geselecteerde contacten apart bijgehouden worden.
- Een knop om de berichten te versturen; alle contacten die (van verschillende SmA's) geselecteerd worden door de gebruiker, moeten het bericht ontvangen.

Back-end

Omdat er verschillende SmA's tegelijk gebruikt kunnen worden voor het versturen van berichten, moest er aan de back-end een extra controle gecreëerd worden. Bij deze controle moeten alle contacten bij de respectievelijke SmA gestopt worden, waarna er per (gevulde) SmA-lijst gecheckt moet worden of de betreffende SmA een berichten-implementatie heeft. Indien dit niet het geval is, dan kunnen de contacten bij de mail-lijst gestopt worden.

Indien een SmA wel een berichten-implementatie heeft, dan moet er eerst vastgesteld worden dat de huidige token voor de SmA nog geldig is. Als dit het geval is, dan kunnen de berichten verstuurd worden naar de betreffende contacten.

In **Figuur 10** staat deze controle afgebeeld doormiddel van een stroomdiagram.



Figuur 10, flow voor berichten controle

Interfaces

In deze, en de vorige sprints zijn er verschillende implementaties toegevoegd. Deze implementaties stuurde tot nu toe objecten terug met alle data die door de SmA teruggestuurd werd. Een deel van deze data mag echter niet doorgestuurd worden naar de client-side. Het gaat hierbij om gevoelige data als de token etc. Indien deze data wel doorgestuurd zou worden, dan kunnen 3^{e} partijen toegang krijgen tot deze informatie.

Daarnaast was er het probleem dat er aan de server-side geen inzicht was in de geretourneerde data. Dit kwam doordat de server niet (meer) kon weten om welke klasse het ging. Bij de front-end was dit geen probleem, omdat javascript het resultaat als generiek object zag.

Om de problemen op te lossen, werden er interfaces toegevoegd aan de plug-in module. Doordat alle implementaties dezelfde (hoofd) interface overerfde, werd er toegang verleend aan de back-end om de (bij de interface specifieke) data in te zien.

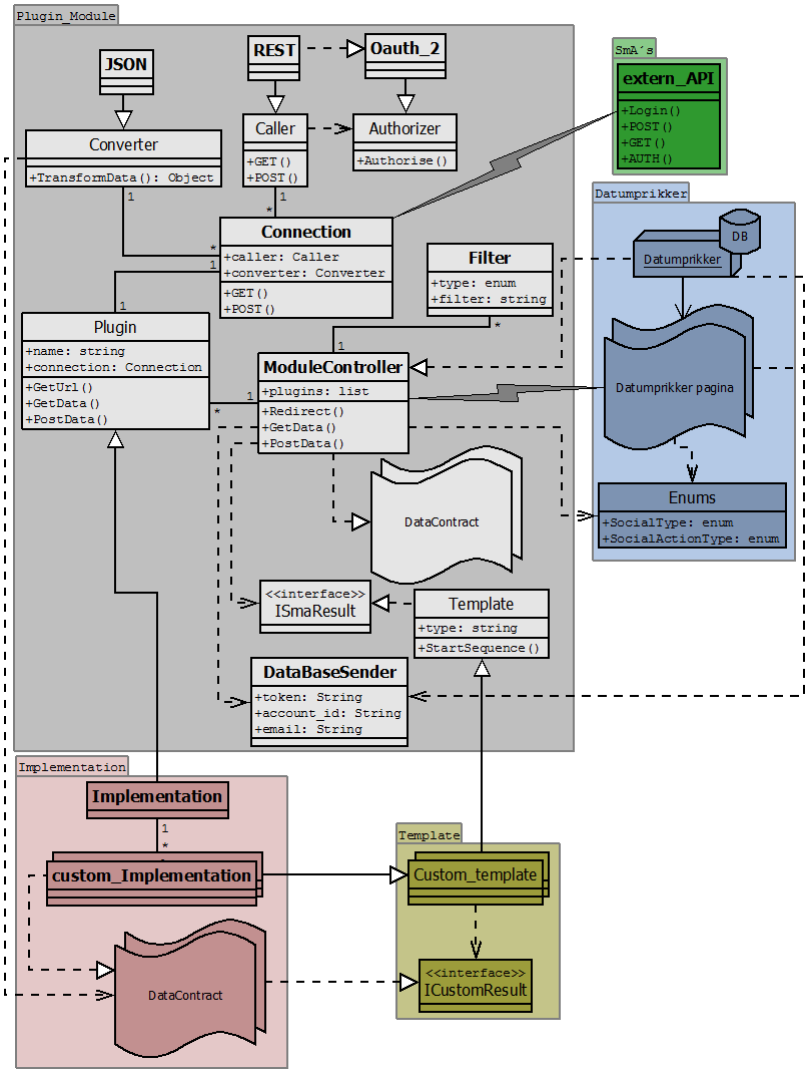
Database object

Het probleem van het weghalen van gevoelige informatie werd hiermee alleen niet goed genoeg opgelost. Daarom is er tevens een database-object klasse aangemaakt. In deze klasse staat alle (gevoelige) data die naar de database gestuurd moet worden. Zodoende wordt deze data dus gescheiden, en niet meer naar de client-side gestuurd.

Ontwerp

Het ontwerp is ondanks alle wijzigingen in de code redelijk hetzelfde gebleven. Bij de vorige sprint was er namelijk al bedacht dat de **DataContracten** voor de **ModuleController** nodig waren. Toch zijn er een aantal wijzigingen gemaakt:

- Er zijn 2 interfaces bijgekomen:
  - **ISmaResult**; deze interface bevat alle (globale) informatie die voor de front-end is bedoeld.
  - **ICustomresult**; deze interface (specifieke) informatie die voor de front-end is bedoeld.
- **DataBaseSender** klasse toegevoegd; hierin staat alle informatie die de back-end (database) van Datumprikker nodig heeft.
- **Enums** toegevoegd; deze geven aan om welke SmA het gaat, en om welke functionaliteit. Deze moesten toegevoegd worden aan de **Datumprikker** package op verzoek van 1 van de stakeholders. Door de **Enums** toe te voegen werd de module (veel) minder een opzichzelfstaand product, en meer een onderdeel van de Datumprikker applicatie. De stakeholder vond echter dat alle enums op één locatie moesten staan.



Figuur 11, ontwerp versie 1.1

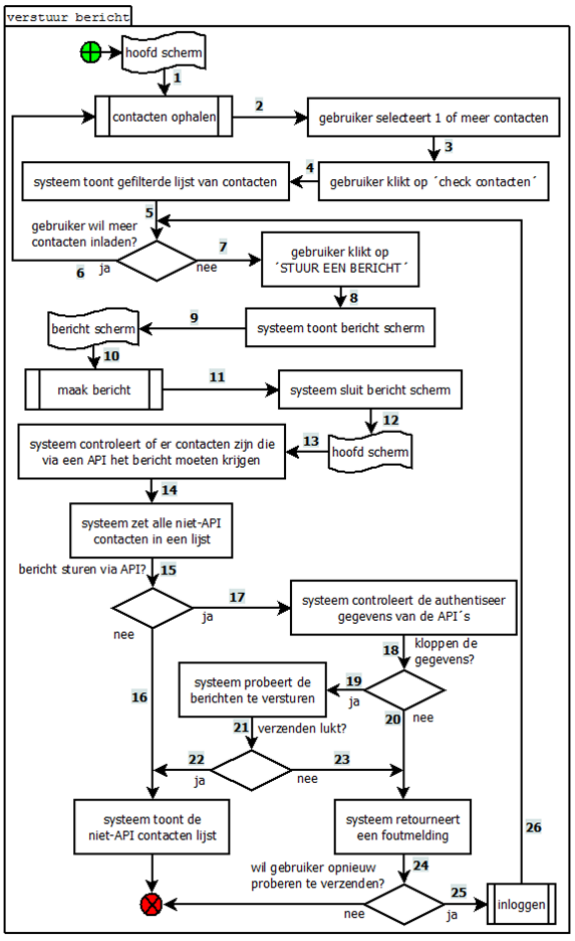
2.4 Proof-of-concept testen

Periode:

Proof-of-concept testen															
week	7					8					9				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Bij de vorige sprint werden de laatste (verplichte) functionaliteiten toegevoegd. Om te controleren of al deze functionaliteiten ook correct werkten, was er voor gekozen om een black-box use-case acceptatie test uit te voeren. Hierbij wordt er gecontroleerd of de acties die een gebruiker uitvoert, ook de correcte resultaten geven. De interne code is hierbij niet van belang, maar eventuele fouten hebben uiteraard wel invloed op de resultaten.

Om de use-cases goed te kunnen testen, moesten eerst alle mogelijke actie-combinaties vastgelegd worden. Dit werd gedaan doormiddel van het opstellen van stroomdiagrammen. In totaal waren er acht stroomdiagrammen nodig om alle mogelijkheden af te kunnen gaan. In **Figuur 12** staat één van de wat ingewikkeldere afgebeeld als voorbeeld.



Figuur 12, stroomdiagram voorbeeld

Op basis van een stroomdiagram als deze kan er bepaald worden welke routes er gevolgd moeten worden om alle mogelijke acties ten minste één keer gehad te hebben. Bij het testen moeten dan alle route combinaties minimaal één keer doorlopen worden.

De mogelijke combinaties voor dit stroomdiagram zijn als volgt[15]:

combinatie	paden
a	S,1,2,3,4,5
b	6,2,3,4,5
c	7,8,9,10,11,12,13,14,15
d	16,E
e	17,18
f	19,21
g	20,24
h	22,E
i	23,24
j	25,26
k	E

Elk pad bestaat uit de nummers tot de eerstvolgende splitsing. Om dus bij het eindpunt te komen, zou bijvoorbeeld de combinaties a, b, c, d achtereenvolgend uitgevoerd kunnen worden.

[15] De nummers van de paden komen overeen met de nummers van het stroomdiagram. De ‘S’ en ‘E’ staan respectievelijk voor het start- en eindpunt.

De acties **contacten ophalen**, **maak bericht** en **inloggen** verwijzen naar de gelijknamige andere stroomdiagrammen. Om het stroomdiagram van **Figuur 12** dus volledig te kunnen doorlopen, moeten ook alle combinaties uit de onderliggende stroomdiagrammen (en eventueel de daarin verwezen stroomdiagrammen) doorlopen worden.

2.4.1 Uitslag

In het onderstaande tabel staat de uitslag van de op het stroomdiagram van **Figuur 12** uitgevoerde test. Bij deze test werden er drie routes gevolgd, en waren er twee SmA combinaties. De reden dat er geen losse SmA's getest werden, komt omdat er bij het versturen van berichten meerdere SmA's tegelijk gebruikt kunnen worden.

Verstuur bericht	Live & Facebook	Google & LinkedIn
Route {a, b, c, d}	succesvol	succesvol
Route {a, c, e, g, j, b, c, e, f, h}	onmogelijk	succesvol
Route {a, c, e, g, j, b, c, e, f, i, k}	onmogelijk	succesvol
Eindresultaat zoals verwacht?	ja	ja
Fout bij pad:	-	-

Wat opvalt bij het bovenstaande tabel, is dat niet alle routes succesvol uitgevoerd konden worden. Toch was het eindresultaat zoals verwacht. Dit komt doordat Live en Facebook beide geen berichten-verstuur functionaliteit ondersteunen. De tweede en derde route kunnen daarom niet uitgevoerd worden.

Andere use-cases

De resultaten van de overige zeven use cases gaven allemaal hetzelfde eindresultaat als bij het voorbeeld. Op basis hiervan kon dus geconcludeerd worden dat het proof-of-concept volgens de requirements functioneerde.

Dit hield in dat de plug-in module geïmplementeerd kon worden in Datumprikker, waarbij er voor de webpagina's het proof-of-concept gebruikt kon worden als blauwdruk.

2.5 Sprint 4: Module gereed maken voor live-gang

Periode:

Sprint 4	Module gereed maken voor live-gang														
week	8					9					10				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Storymap:

Code	Requirement
Nb1	De plug-in module moet volgens de use cases handelen.
No5	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om aan de data van de SmA's te komen.

De focus van deze sprint ligt op het voorbereiden voor de live-gang.

2.5.1 Requirements

Voordat deze sprint begon, zijn de volgende requirements toegevoegd:

id	Requirements
Nb1	De plug-in module moet volgens de use cases handelen.
No5	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om aan de data van de SmA's te komen.

2.5.2 Keuzes en problemen

Hoewel het proof-of-concept volgens de (tot nu toe opgestelde) requirements voldeed, was de module nog niet gereed om direct in Datumprikker toe te voegen. Hiervoor waren er twee redenen.

Allereerst zou Datumprikker bij de komende (nieuwe) live-gang niet volledig gebruik maken van alle functionaliteiten van de plug-in module. Dit kwam doordat er behoorlijk wat interne veranderingen gemaakt moesten worden om de nieuwe functionaliteiten te kunnen implementeren.

Om toch de module mee te nemen met de live-gang, werd er voor gekozen om alleen de functionaliteit van contacten importeren voor Google en Live te implementeren. Dit zou de kleinste impact hebben op Datumprikker.

De andere reden was dat de medewerker die de module moest inbouwen een aantal wensen had.

Specifieke methodes

Eén van deze wensen was om specifiekere methodes te gebruiken om de plug-in module te benaderen. Hierdoor werd requirement No5 niet nageleefd, maar het werd wel veel makkelijker voor Webbeat om de plug-in module te benaderen. Er werd namelijk voor elke functionaliteit één methode gemaakt, waarbij alleen maar variabelen ingevuld hoefde te worden die ook echt nodig waren.

Deze methodes riepen intern simpelweg de originele methodes aan, waardoor er geen verdere aanpassingen gemaakt hoefde te worden binnen de module. Er is hier dus gebruik gemaakt van een façade design pattern. Met dit design pattern werd het data contract dat in de vorige sprint opgesteld was, overbodig gemaakt. Dit is daarom dan ook verwijderd.

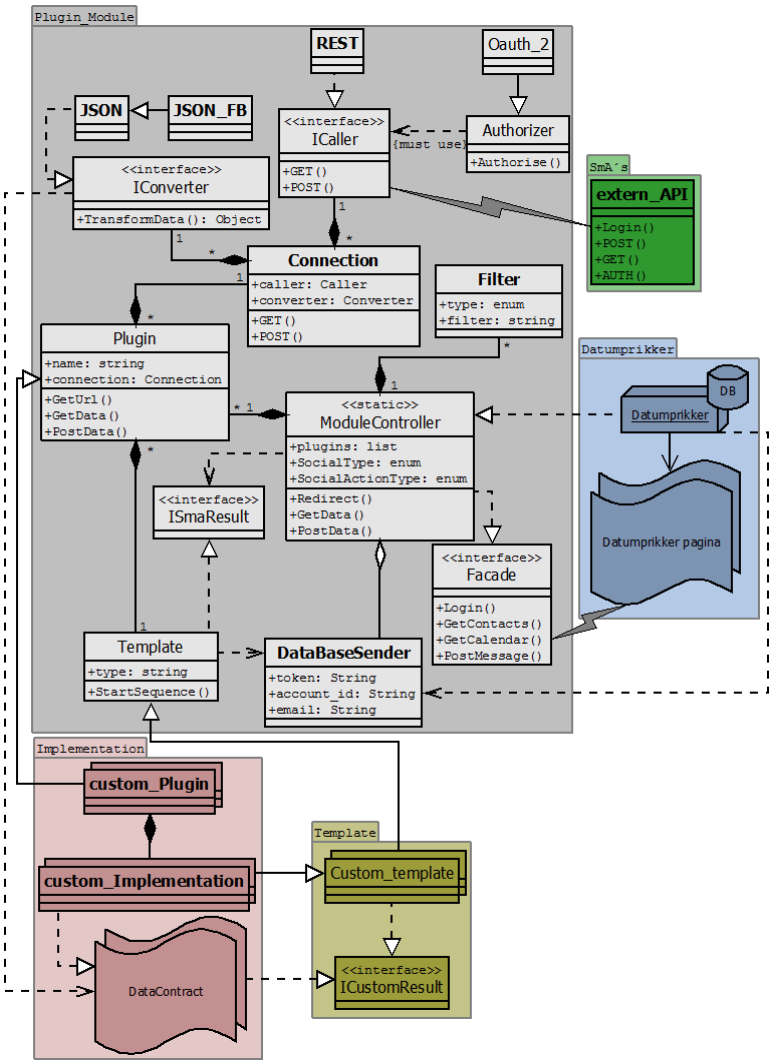
Ontwerp

Deze wijzigingen zorgde uiteraard weer voor aanpassingen in het ontwerp, te zien in **Figuur 13**. Hierin is ook te zien dat er nog een aantal kleine wijzigingen zijn doorgevoerd.

Zo is de klasse JSON_FB (eindelijk) toegevoegd aan het ontwerp. Deze klasse was altijd al aanwezig binnen de code (sinds Facebook, **H2.1.1**), maar was vergeten toe te voegen aan het ontwerp. Met deze klasse wordt de Facebook-bug opgelost.

Een andere wijziging is de Authorizer klasse. In plaats van een dependency, is er nu een restrictie afgebeeld. Op code-niveau veranderde er niets, maar het is op deze manier wel een correcte(re) weergave.

Als laatste is de enum die in de vorige versie was toegevoegd, nu na overleg met de stakeholders, verplaatst naar de klasse **ModuleController**. Door de enums binnen de module te houden, is deze volledig opzichzelfstaand, en dus ook voor andere producten inzetbaar.



Figuur 13, ontwerp versie 1.2

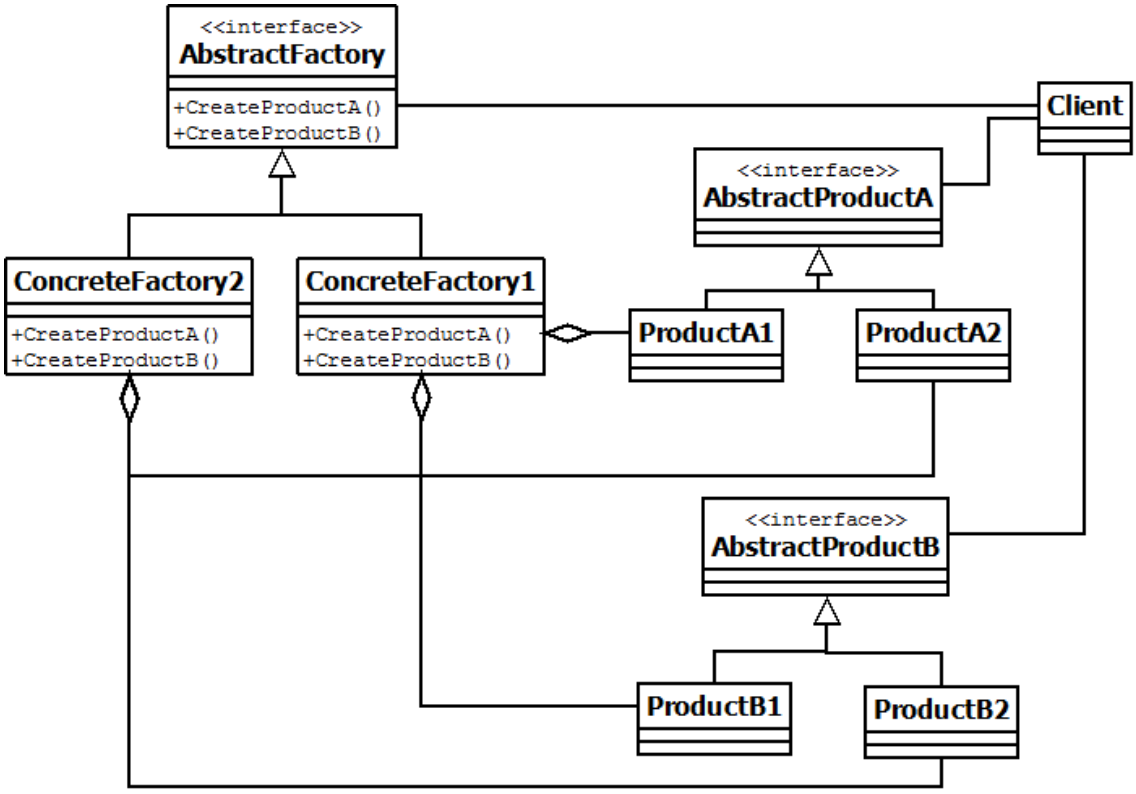
Design patterns

Naast het eerder genoemde façade pattern, worden er binnen de plug-in module nog meer design patterns aangehouden, zoals bijvoorbeeld de abstract factory.

Met de abstract factory wordt het mogelijk gemaakt om tijdens run-time verschillende implementaties aan te maken, zonder deze te specificeren. Dit wordt gedaan door een interface aan te roepen, die op basis van de input bepaalt welke factory er gebruikt zal worden.

Er zijn dus meerdere factories binnen het pattern, die elk een eigen set aan objecten aan kan maken. De aanroeper (client) van de interface weet echter niet welke factory (en welke (concrete) objecten) er aangeroepen worden. Dit maakt ook niet uit, aangezien de aangemaakte objecten op hun beurt weer allemaal overerven van een andere interface. De client weet dus alleen af van die betreffende interface(s), en van de interface om deze aan te maken.

In **Figuur 14** is de globale versie te zien van een abstract factory. Hierin is te zien dat er 2 factories zijn, die elk een andere implementatie hebben van de interfaces **AbstractProductA** en **AbstractProductB**. Als de client een object wil aanmaken van één van deze interfaces, dan moet deze de bijbehorende methodes van de **AbstractFactory** aanroepen.

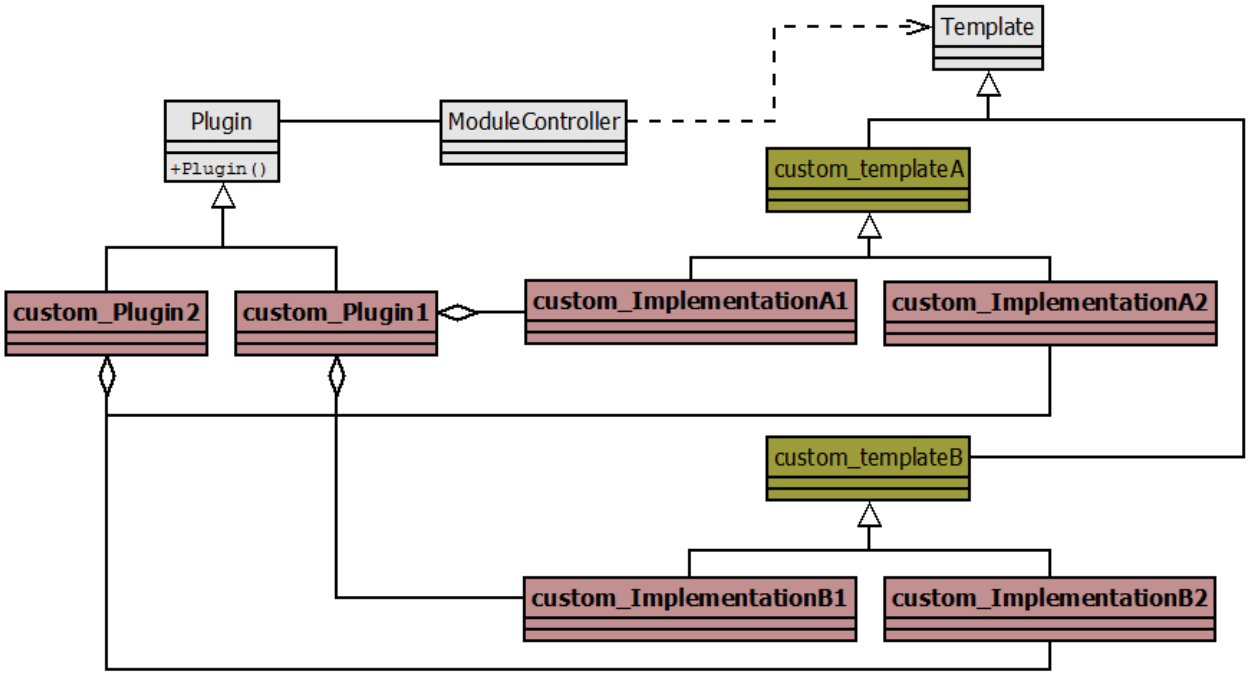


Figuur 14, abstract factory (globaal)[16]

[16] Bron: http://www.lepus.org.uk/ref/companion/AbstractFactory.xml

Plug-in module

Het pattern van **Figuur 14** wordt binnen de plug-in module gebruikt voor de implementaties van de plug-ins. In **Figuur 15** is te zien hoe dit pattern binnen het ontwerp gebruikt wordt.



Figuur 15, abstract factory (gerealiseerd)

Om de gelijkenis overzichtelijk te houden, zijn er net zoveel factories en objecten getoond als bij de globale versie. Binnen de module zijn er echter 4 **custom_Plugin**'s[17], en 4 **custom_template**'s[18] aanwezig. Daarnaast zijn alle onnodige methodes, variabelen en klassen van het ontwerp weggelaten.

De gerealiseerde versie verschilt op een aantal punten met de globale versie:

- De **ModuleController** klasse (client) heeft niet directe toegang tot de templates (AbstractProduct), maar via de **Template** klasse. Dit is omdat de **ModuleController** niet hoeft te weten welk type object er aangemaakt wordt.
- Er zijn geen aparte methodes voor het creëren van de objecten. In plaats daarvan wordt dit binnen de constructor van de **Plugin** klasse gedaan.
- De **Plugin** klasse (AbstractFactory) en de **custom_template**'s (AbstractProduct) zijn geen interfaces, maar abstracte klassen.

Interface vs abstracte klasse

Interfaces en abstracte klassen lijken veel op elkaar, maar ze verschillen op één belangrijk punt. Een interface heeft namelijk geen concrete inhoud. Het is puur een stuk informatie dat aangeeft waar objecten aan moeten voldoen indien ze de interface implementeren. Hierdoor kan een aanroepende klasse altijd bij de informatie die in de interface staat opgesteld, zonder de (inhoud van de) implementerende klasse te hoeven kennen.

[17] Eén voor elke SmA: Facebook, Google, LinkedIn en Microsoft Live
[18] Eén voor elke functionaliteit: Inloggen, Contacten, Kalender, Berichten

Een abstracte klasse is daarentegen een daadwerkelijke klasse. Dit betekend dat deze concrete variabelen en functies kan bevatten. Doordat de klasse abstract is, kan er echter (net zoals bij de interface) geen instantie van gemaakt worden. Dit betekend dat er een klasse van moet overerven om gebruik te kunnen maken van de functionaliteit.

De reden dat er bij de concrete versie van het design pattern gekozen is voor abstracte klassen in plaats van interfaces, is omdat de betreffende klassen ook daadwerkelijk abstracte klassen zijn. Ze hebben allemaal concrete inhoud, en de methodes worden door de overervende klassen gebruikt, in plaats van (alleen) geïmplementeerd.

Gebruikte design patterns binnen module

In totaal zijn er 5 design patterns gebruikt:

- Abstract factory
- Adapter
- Strategy
- Façade
- Proxy

Deze patterns zijn allemaal op een zelfde manier geïntegreerd zoals bij het abstract factory pattern is uitgelegd, en staan allemaal uitgewerkt in het **Ontwerp** document.

2.6 Sprint 5: Kalender posten

Periode:

Sprint 5	Kalender posten														
week	10					11					12				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Storymap:

Code	Requirement
F1e	Er moet een afspraak toegevoegd kunnen worden aan de kalender van de SmA's.
No6	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om data te versturen naar de SmA's.

De focus van deze sprint lag bij het implementeren van de functionaliteit om kalender data te versturen naar de SmA's. Deze functionaliteit zou alleen geïmplementeerd worden als er genoeg tijd voor over was. Aangezien (een deel van) de module pas een paar weken later live zou gaan, en het project daarnaast pas net over de helft van de totale tijd zat, was dit zeker het geval.

2.6.1 Requirements

Voordat de sprint startte, is de volgende requirement opgesteld:

id	Requirements
No6	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om data te versturen naar de SmA's.

2.6.2 Keuzes en problemen

Aan de webpagina werd een hoop veranderd. Waar er eerst een hoop variabelen in Javascript stonden, werden deze nu vervangen door elementen als invoervelden. Dit werd gedaan zodat het mogelijk werd om van de (door de gebruiker aangemaakte data) 1 datum optie te kunnen selecteren, en hier dan nog andere informatie zoals een afspraakonderwerp aan toe te voegen. Daarnaast moest het mogelijk worden om kalenders van de verschillende SmA's te selecteren, zodat er in meerdere kalenders tegelijk een afspraak geplaatst kon worden.

Door deze wijzigingen door te voeren, kon er een datum (met bijbehorende variabelen zoals een titel etc.) opgesteld worden, en naar de module gestuurd worden.

Module

Om iets met de datum te kunnen doen, moest er een nieuwe methode aangemaakt worden binnen de façade die de variabelen kon ontvangen, en daarna doorsturen naar de bijbehorende implementatie(s). Deze implementaties moesten de gegevens kunnen versturen naar de volgende SmA's:

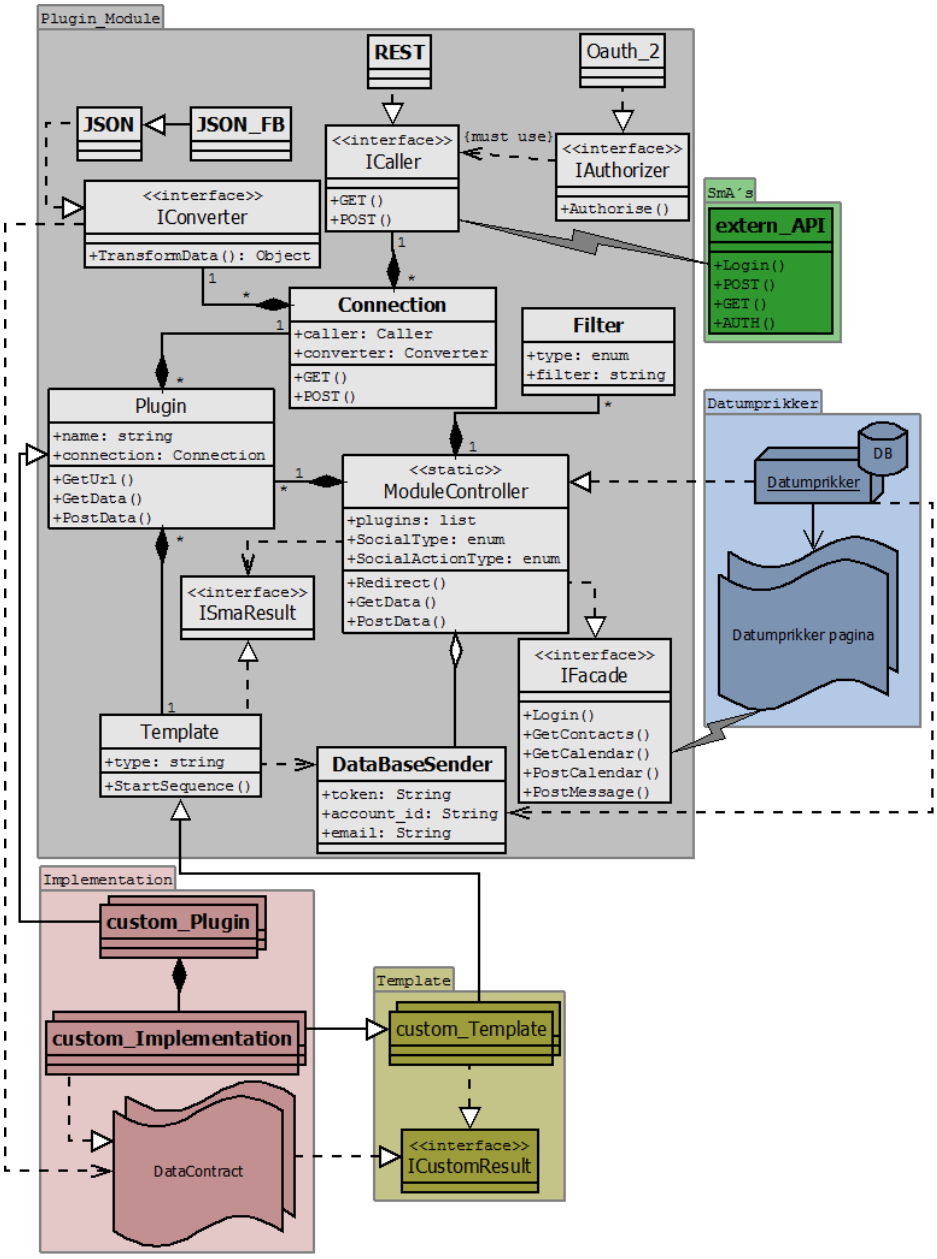
- Microsoft Live
- Google (Calendar)
- Facebook

SmA's

LinkedIn heeft geen kalender functionaliteit, dus hier hoefde dan ook geen implementatie voor gemaakt te worden. Microsoft was de enige SmA die dit keer voor problemen zorgde. Dit kwam opnieuw door slechte documentatie. Bij het posten van data zijn er namelijk weer andere regels dan bij het opvragen van data. Hierdoor werd er in eerste instantie weer steeds de '415' foutmelding geretourneerd.

Ontwerp

De enige wijziging die bij het ontwerp is doorgevoerd is de wijziging in de façade. Hiermee is het uiteindelijke ontwerp er als volgt uit komen te zien:



Figuur 16, uiteindelijk ontwerp

2.7 Extra onderzoek

Na de 5e sprint zijn er een aantal onderwerpen extra onderzocht voor Webbeat. De opdrachtgever wilde namelijk weten of er nog extra aanpassingen gemaakt moesten worden aan de module en/of Datumprikker om via de (toekomstige) app's gebruik te maken van de SmA's.

Daarnaast wilde de opdrachtgever weten welke aanpassingen er nog gedaan moesten worden aan Datumprikker om volledig gebruik te kunnen maken van de plug-in module. Hierbij moet gedacht worden aan aanpassingen aan de database, webpagina's, etc.

De resultaten van deze onderzoeken zijn in het **Onderzoek Social Media** document mee opgenomen.

2.8 Monitoring

Periode:

monitoring															
week	14					15					16				
dag	m	d	w	d	v	m	d	w	d	v	m	d	w	d	v

Het was de bedoeling dat er direct vanaf het live gaan gemonitord zou worden. Helaas was dit niet mogelijk vanwege verschillende problemen en andere factoren die buiten de scope optraden. Hierdoor is er de week erop pas begonnen met de monitoring.

2.8.1 Voorbereiding

Om de plug-in module qua gebruik te kunnen monitoren, moest deze uiteraard eerst op de live-omgeving gezet worden. Dit werd gedaan door de medewerkers van Webbeat, die naast de plug-in module ook gelijk diens eigen (hoofd)project live zetten.

Omdat de plug-in module een apart onderdeel is binnen de Datumprikker applicatie, kostte het vrijwel geen moeite om deze te integreren met de applicatie. Er hoefde namelijk alleen maar de dll toegevoegd te worden, en daarnaast een paar kleine aanpassingen en verwijzingen.

Bij het live gaan is echter niet alle functionaliteit van de plug-in module in gebruik genomen. Dit kwam doordat een aantal functionaliteiten grotere wijzigingen betekende binnen de Datumprikker applicatie, omdat dit de bestaande functionaliteiten zou wijzigen en/of uitbreiden. Er is daarom gekozen om alleen de mogelijkheid om Google en Microsoft Live contacten in te laden, toe te voegen. Deze functionaliteit maakt echter wel van een groot deel van de plug-in module gebruik, dus het kon goed laten zien of de module correct werkte, en of mensen er veel gebruik van willen maken.

2.8.2 Fouten

Tijdens de monitoring was er slechts één keer een fout opgetreden. Dit was in het begin van de periode, en kwam doordat iemand contacten probeerde in te laden uit Google, terwijl er bij dat account geen contacten bestonden.

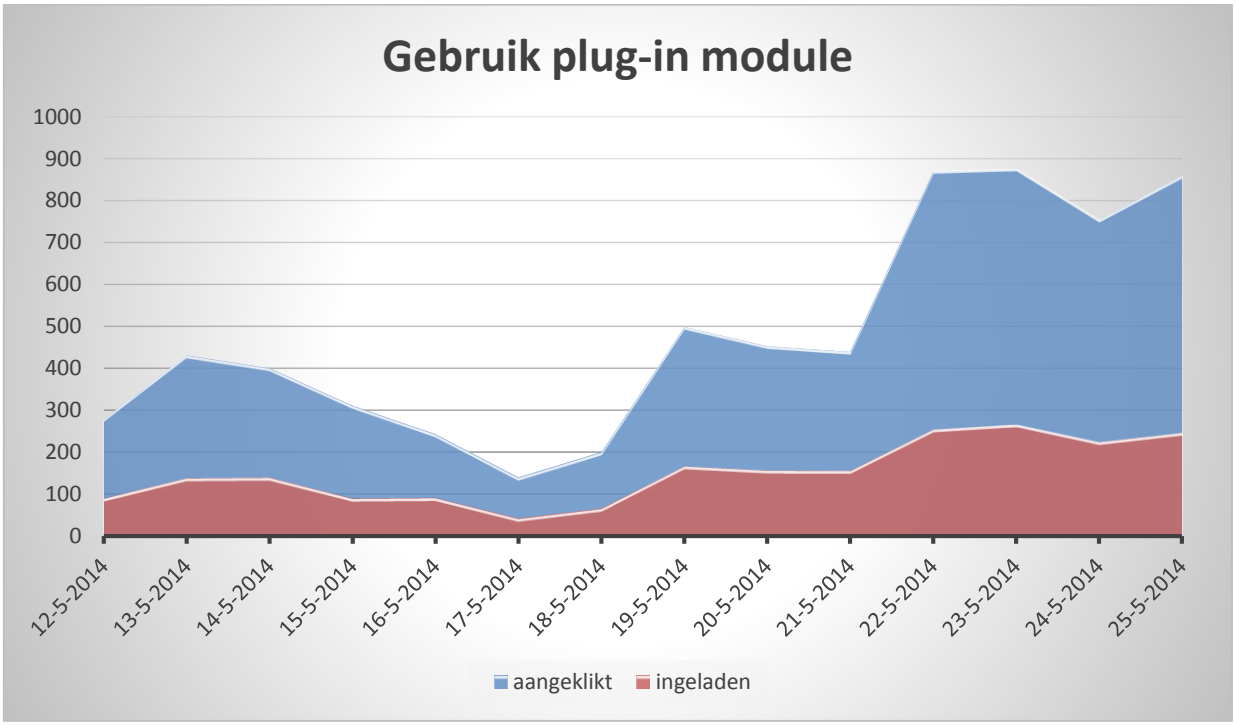
Hoewel er in de code rekening gehouden werd met lege resultaten, was er niet voldoende rekening gehouden met deels ingevulde resultaten. In dit geval stuurde de Google SmA namelijk wel een resultaat, maar de contacten lijst was wel leeg. Gelukkig ving de module keurig de fout op, waardoor de gebruiker er nooit iets van gemerkt heeft. Er werd namelijk een lege contactenlijst getoond, en dat was zonder de fout ook gebeurd. De fout kon snel opgelost worden, en daarna trad deze (of andere fouten) niet (meer) op.

Aangezien er in de periode van twee weken slechts eenmalig een fout optrad, en deze van zeer geringe zwaarte was, kon er geconcludeerd worden dat de plug-in module correct werkt bij hete ophalen van contacten van Google en Microsoft Live.

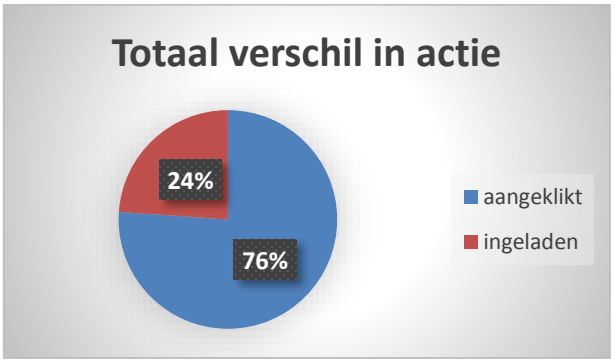
2.8.3 Resultaten

In **Figuur 17** is goed te zien hoe vaak er gebruik gemaakt is van de plug-in module tijdens de twee weken van het monitoren. Wat meteen opvalt, is dat mensen wel vaak voor de optie om contacten in te laden kiezen, maar dat er maar weinig mensen zijn die daadwerkelijk bij de SmP inloggen en de contacten inladen. Dit gedrag is niet gemakkelijk te verklaren, aangezien er hier vele redenen voor kunnen zijn. Zo beweerde een van de medewerkers van Webbeat dat een hoop mensen niet realiseren dat ze rechten moeten geven voor het gebruik van de SmA functionaliteiten.

Als er naar **Figuur 18** gekeken wordt, dan is het verschil in gebruik nog duidelijker te zien. Van alle mensen die voor de optie kozen om contacten in te laden, zetten maar 24% (= 2119x) door. Driekwart van de gebruikers logde dus niet in bij de SmP en/of weigerde de rechten te geven.



Figuur 17, gebruik plug-in module



Figuur 18, verschil in actie

Uitleg grafieken

- "aangeklikt" betekend dat een gebruiker kiest om bij één van de SmP's contacten in te gaan laden.
- "ingeladen" betekend dat een gebruiker na het aanklikken inlogt bij de betreffende SmP, en toestemming geeft aan Datumprikker om de contacten in te laden. Hierna toont Datumprikker de ingeladen contacten aan de gebruiker.

3 Evaluatie

Nu het afstudeer project afgerond is, kan er geëvalueerd worden over het gehele project.

3.1 Proces

Omdat er zulke goede communicatie was, was het ook gemakkelijk om aan de manier van werken van Webbeat te wennen. Deze communicatie kwam vooral van pas rond de live-gang, aangezien er toen goed samengewerkt moest worden om een goede koppeling te kunnen krijgen tussen de plug-in module en Datumprikker.

3.1.1 Software ontwikkelmethode

De software ontwikkelmethode die Webbeat aanhoudt past ook echt bij het bedrijf. Er wordt regelmatig (1 op 1) overlegd, en overal zijn er whiteboards etc. waarop verschillende projecten uitgewerkt worden/zijn.

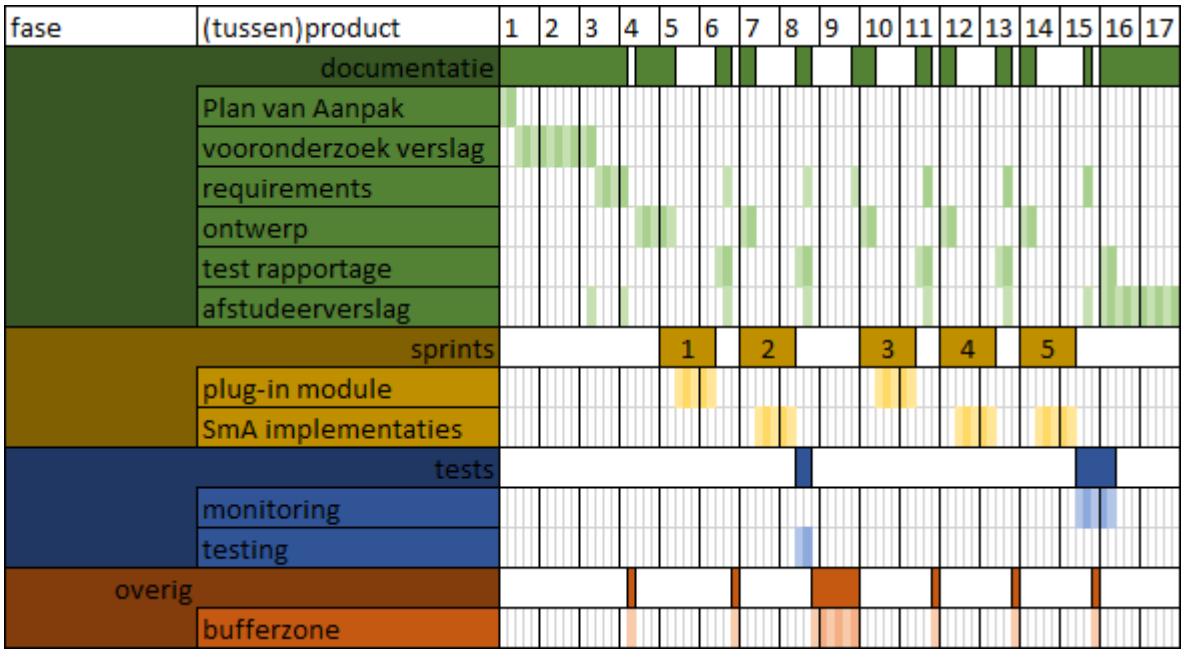
Hoewel er voor dit project andere methodes gebruikt hadden kunnen worden, hadden deze zeer waarschijnlijk een te grote impact gehad op de huidige manier van werken. Het was daarom ook een verstandige keus om voor dezelfde manier van werken te kiezen (met de voor de afstudeeropdracht-gerichte aanpassingen).

3.1.2 Planning

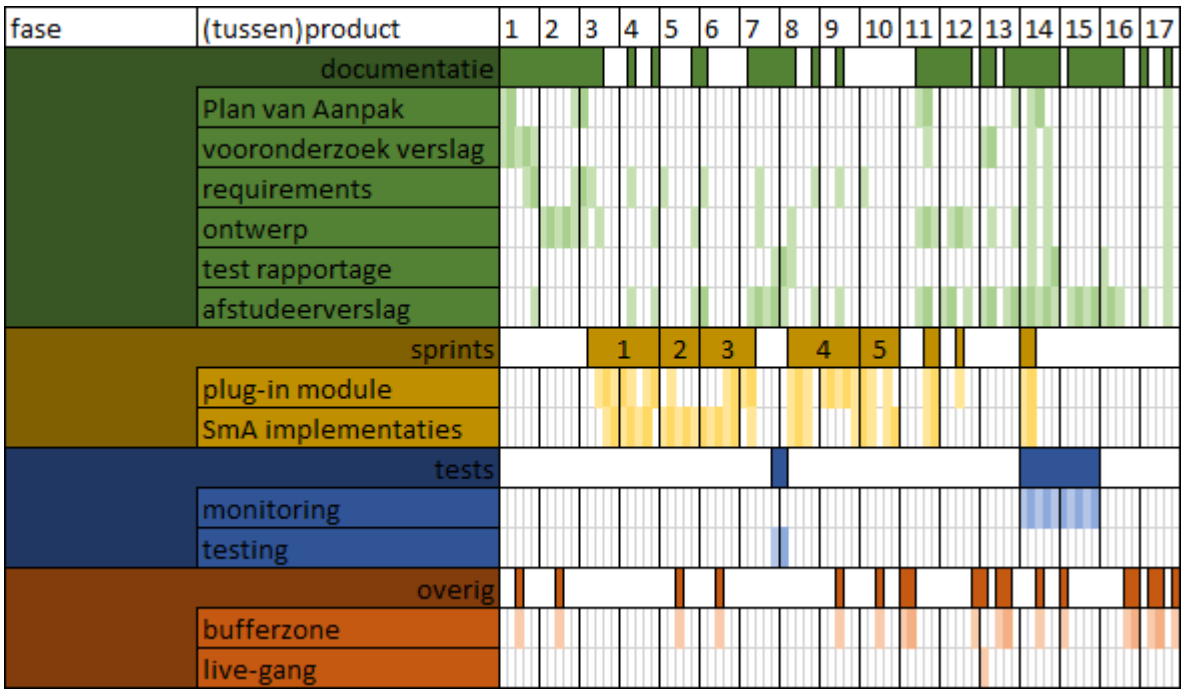
De oorspronkelijke planning bleek al snel te ruim ingepland. Maar dit was geen nadeel voor het project. Het zorgde er namelijk voor dat er voor alle onderdelen genoeg tijd was, en dat er uiteindelijk zelfs tijd over was voor wat extra dingen.

In **Figuur 19** en **Figuur 20** staan de originele en uiteindelijke planning onder elkaar. Hierdoor is het verschil goed te zien.

Bij een volgend project dat met sprints werkt zou ik zeer waarschijnlijk een soortgelijke planning aanhouden. Zeker de bufferzones die van te voren inberekend waren, bleken achteraf zeer nuttig. Sprints konden hierdoor namelijk uitlopen zonder de planning echt in gevaar te brengen.



Figuur 19, originele planning



Figuur 20, uiteindelijke planning

3.2 Plug-in module

Het eindproduct (de plug-in module) is op het moment van schrijven, in zoverre afgerond dat er hier geen aanpassingen meer aan hoeven gemaakt te worden om geïmplementeerd te kunnen worden bij Datumprikker (of een ander product). Om volledig gebruik te kunnen maken van alle functionaliteiten van de plug-in module moeten de producten die er gebruik van willen maken zelf waarschijnlijk wel eventuele aanpassingen ondergaan.

3.3 Andere opgeleverde producten

Naast het hoofdproduct, zijn er in de loop van het afstudeertraject uiteraard nog andere producten opgeleverd. Deze worden hier kort besproken.

3.3.1 Plan van aanpak

Het opstellen van het plan van aanpak zorgde niet voor echte problemen. Dit kwam allereerst doordat het afstudeerplan aangehouden kon worden. Daarnaast waren er geen (verdere) lastige keuzes te maken voor dit product.

3.3.2 Onderzoeksrapport

De verschillende onderzoeken die zijn gedaan waren op zich goed uit te voeren. Wel bleek er na het vooronderzoek naar de SmA's later veel info te ontbreken en/of incorrect te zijn. Doordat er pas bij het ontwerp, en vooral tijdens het bouwen, duidelijk werd dat de informatie niet klopte, zorgde dit voor onnodig extra (onderzoek) werk.

Toch zou ik bij een volgend project het vooronderzoek niet veel anders aanpakken. Het is zinloos om je in elk feit te gaan verdiepen. Daarnaast komt het toch vaker voor dat er tijdens het bouwen nieuwe of andere inzichten komen waardoor er ergens specifiek op gezocht moet worden.

3.3.3 Requirements

Tijdens dit project zijn er redelijk wat requirements bijgekomen na de initieel opgestelde. Dit kwam uiteraard doordat er bij het begin van elke sprint gekeken werd wat er in die periode gedaan zou worden. Hierdoor kwamen er specifieke eisen naar boven.

Aangezien dit vrij normaal is bij een Agile methode, zou dit bij een volgend project (dat Agile aanhoudt) zeer waarschijnlijk weer gebeuren.

3.3.4 Ontwerp

Zoals was te verwachten bij een project met sprints, werd het ontwerp regelmatig aangepast. Dit kwam omdat er steeds nieuwe functionaliteit bij kwam.

Wat wel redelijk opvallend is, is dat het begin ontwerp qua globaal idee niet eens zo heel veel verschilt van de uiteindelijke versie. Hieruit kan opgemaakt worden dat er vanaf het begin al meteen een goed idee was over hoe de module eruit zou komen te zien.

Aangezien een ontwerp voor elk project anders is, is er niet veel te zeggen over hoe de opgedane kennis gebruikt zou kunnen worden bij een volgend project. Wel zal er zeer waarschijnlijk een UML(achtige) ontwerp methode gebruikt worden, aangezien dit de standaard is.

3.3.5 Test rapportage

De uitvoering van de acceptatie tests was niet veel anders dan bij eerdere uitgevoerde projecten. Dit maakte het dan ook niet zeer uitdagend. Aangezien testen uiteraard wel belangrijk is, zal er bij een volgend project zeer waarschijnlijk een vergelijkbaar rapport opgesteld worden.

Wat echter wel interessant was, was de monitoring. Dit was namelijk de eerste keer dat een product van mij (tijdens een project) live ging voor een groot publiek. Het feit dat er tijdens de monitoring-fase slechts één fout optrad (die toch geen invloed had op de uitkomst), en dat er meer dan 2000 mensen succesvol gebruik maakte van de plug-in module, was een mooie afsluiting voor dit project.

3.4 Competenties

In de hier onderstaande paragrafen staan alle competenties die binnen dit project gebruikt zijn. Deze competenties zijn allemaal zelfstandig als taak-rol uitgevoerd. Dit houdt in dat de zwaarte van de competenties tussen de 2 en 4 kan liggen.

Context		Taakrol		
		Geleid	zelfstandig	Sturend
	simpel	1	2	3
	lastig	2	3	4
	complex	3	4	5

Figuur 21, competentie matrix [19]

3.4.1 Uitvoeren analyse door definitie van requirements(1.4)

Hoewel er geen grote hoeveelheden requirements opgesteld hoefde te worden gedurende dit project, waren er wel veel verschillende bronnen waaruit deze eisen naar voren moesten komen (zoals in H**1.5.4** staat beschreven). Daarnaast kwamen er bij elke sprint nieuwe requirements bij (H**2.1.1**, H**2.2.1**, H**2.3.1**, H**2.5.1** en H**2.6.1**), waardoor er continu versiebeheer plaats moest vinden.

Ook waren er op verschillende momenten tegenstrijdige meningen bij de stakeholders over de eisen van de module (zoals beschreven in H**2.2.2**-Email-adressen Facebook, H**2.2.2**-**Offline token**, H**2.3.2**-Ontwerp en H**2.5.2**-**Ontwerp**), en klopten de eisen van de SmP's ook niet altijd (beschreven in H**2.1.2**-**Facebook**).

Al deze punten tezamen zorgde voor een complexiteit van 'complex', waardoor het niveau 4 behaald is.

3.4.2 Ontwerpen systeemdeel (3.2)

Doordat er in sprints gewerkt werd, werd het ontwerp ook meerdere malen aangepast (zoals beschreven in H**2.1.2**-Ontwerp, H**2.2.2**-Ontwerp, H**2.3.2**-Ontwerp, H**2.5.2**-Ontwerp, H**2.6.2**-Ontwerp). Dankzij het (uiteindelijke) ontwerp is het tevens mogelijk om de plug-in module voor andere producten/projecten te gebruiken (zoals beschreven in H**2.5.2**-**Ontwerp**). Het is dus een volledig opzichzelfstaande module geworden die met 4 verschillende SmA's kan communiceren (zoals bewezen in H**2.4**).

Om tot dit ontwerp te komen, zijn verschillende design patterns gebruikt, zoals beschreven in H**2.5**-**Design patterns**.

Al deze punten tezamen zorgde voor een complexiteit van 'complex', waardoor het niveau 4 behaald is.

[19] Bron: Beroepstaken van de opleiding Informatica – Academie voor ICT & Media (juni 2009, versie 1.1)

3.4.3 Bouwen applicatie (3.3)

Het is wat lastiger om te bewijzen dat er voldaan is aan deze competentie, aangezien de code van de plug-in module niet besproken wordt binnen dit verslag. Wat echter wel besproken kan worden, zijn de uitslagen van de uitgevoerde testen (beschreven in H**2.4**), en de resultaten uit de monitoring-sessie (beschreven in H**2.8**).

Uit deze rapporten komt naar voren dat de plug-in module correct werkt, en reeds dagelijks gebruikt wordt door de gebruikers van Datumprikker. Hieruit kan opgemaakt worden dat dit project succesvol is afgerond.

De zwaartepunten binnen dit project lagen bij de volgende punten:

- De functionaliteiten van verschillende SmP's moesten samengevoegd worden tot één product (beschreven in H**1.1** en H**1.5.3**).
- Er moest rekening gehouden worden met een modulaire structuur (beschreven in H**1.1** en **Bijlage 1: Storymap**).
- Er werd gewerkt in een aparte ontwikkelomgeving (beschreven in H**1.3**), waarna de plug-in module als dll werd geïntegreerd met de live-omgeving (beschreven in H**2.8.1**).

Al deze punten tezamen zorgde voor een complexiteit van 'lastig', waardoor het niveau 3 behaald is.

3.4.4 Uitvoeren van en rapporteren over het testproces (3.5)

Hoewel deze competentie van te voren niet is opgenomen in het afstudeerplan, bleek tijdens het project dat er hier toch meer complexiteit bij kwam kijken dan eerst gedacht werd. Vandaar dat deze competentie hier toch besproken wordt.

Tijdens dit project is er twee keer een rapport opgesteld (beschreven in H**2.4** en H**2.8**), die beide concluderen dat de plug-in module volgens de opgestelde requirements functioneert.

Met name bij het rapport van H**2.4** is er gebruik gemaakt van een testontwerp-techniek. Daarnaast werd er rekening gehouden met herhaalbaarheid van de testen.

Al deze punten tezamen zorgde voor een complexiteit van 'lastig', waardoor het niveau 3 behaald is.

Bijlages

De volgende bijlages zijn toegevoegd aan dit document:

1. Storymap; hier staat de storymap die opgesteld is voor dit project in A3-formaat afgebeeld.
2. Begrippenlijst; hier staan alle begrippen (en eventuele bronnen) beschreven en uitgelegd die binnen het project gebruikt worden.

Daarnaast zijn er een aantal externe documenten (de (tussen)producten) die in dit verslag besproken zijn:

- **Onderzoek Social Media**
- **Ontwerp**
- **Plan van aanpak**
- **Requirements**
- **Test rapportage**

Deze documenten zijn samengevoegd tot één extern document, genaamd **(Tussen)producten**.

F1a		F1b	F1d	Nb1	F2e
Er moet ingelogd kunnen worden met behulp van de SmA's.		Er moeten contacten van de gebruiker opgehaald kunnen worden met behulp van de SmA's.	Er moeten berichten naar bekende accounts van de gebruiker gestuurd kunnen worden met behulp van de SmA's.	De plug-in module moet volgens de use cases handelen.	Er moet een afspraak toegevoegd kunnen worden aan de kalender van de SmA's.
100%		100%	100%	100%	100%
F2	F3	F1c	No4	No5	No6
Er moet een module gebouwd worden die plugins kan bevatten.	De plugins dienen elk met 1 SmA te kunnen communiceren.	Er moet agenda data van de gebruiker opgehaald kunnen worden met behulp van de SmA's.	De website(s) van Datumprikker moeten weten wat voor type object (van No2) er ontvangen zal worden.	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om aan de data van de SmA's te komen.	De website(s) van Datumprikker moeten per implementatie maximaal 1 methode gebruiken om data te versturen naar de SmA's.
F4	F5a	No2			
De plug-ins moeten JSON accepteren van de SmA's.	De plug-ins moeten met REST om kunnen gaan.	De requirements F2a, F2b, F2c, F2d, F2e moeten los van elkaar geïmplementeerd kunnen worden.			
F5c	F6	No3			
De plug-ins dienen met Oauth V2 om kunnen gaan.	De van de SmA's ontvangen data moet omgezet worden naar een generiek object.	Het generieke object van No2 zal per implementatie moeten verschillen.			
No1	Nv5				
De module moet een opzichzelfstaande structuur hebben.	De authenticatie dient gedaan te worden met Oauth V2				
F7	F8				
Templates moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.	Plugins moeten toegevoegd kunnen worden aan de module zonder code te moeten toevoegen aan de module zelf.				
Nv1	Nv2				
De module moet gekoppeld zijn aan een bestaand developer account van de social media.	De module moet geregistreerd zijn bij de social media.				
Nv3					
De module moet in de developers console geregistreerd worden binnen een bestaand account van de social media.					

Bijlage 2: Begrippenlijst

	Begrip	Uitleg	Bron / meer informatie
A	abstract factory	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.1.	http://www.lepus.org.uk/ref/companion/AbstractFactory.xml
	adapter	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.2.	http://www.lepus.org.uk/ref/companion/Adapter(Object).xml
	admin	**administrateur**, *vakterm*; iemand verantwoordelijk is voor alle rechten etc. die gebruikers bij een systeem hebben.	http://nl.wikipedia.org/wiki/Administrateur
	aggregaat	*vakterm*, **UML**; een klasse die onderdeel(aggregaat) van een andere klasse is. De aggregaat kan zonder de ander klasse bestaan.	
	agile	*vakterm,* **Scrum** | **XP**; zie **plan van aanpak**, hoofdstuk 3.1.1.	ISBN: 978-0-471-20282-0
	Agile Manifesto	*vakterm*, **Agile**; Hierin staan de richtlijnen en principes die bij Agile horen.	
	API	**application programming interface,** *vakterm*; een verzameling definities op basis waarvan een computerprogramma kan communiceren met een ander programma of onderdeel (meestal in de vorm van libraries).	http://www.3scale.net/wp-content/uploads/2012/06/What-is-an-API-1.0.pdf
	app	**(mobile) application,** *vakterm*; een softwareapplicatie die ontworpen is om te draaien op een elektronisch handapparaat.	http://en.wikipedia.org/wiki/Mobile_application_software
	array	*vakterm*; een lijst van elementen, gebruikt in de IT.	http://en.wikipedia.org/wiki/Array_data_structure
B	backlog	*vakterm*, **Scrum** | **user story**; een lijst van user stories.	
	bi-directioneel	*vakterm*, **UML**; een associatie waarbij beide klassen van elkaar weten wat ze inhouden.	
	black-box	*vakterm*; zie **test rapportage**, hoofdstuk 2.1.1.1.	ISBN: 978-9-012-11883-5
	boolean	*vakterm*; een booleaanse uitdrukking, gebruikt in de IT. Een boolean kan maar 2 waarden hebben: true(waar) of false(niet waar)	http://msdn.microsoft.com/en-us/library/system.boolean.aspx
C	client	*vakterm*, **design pattern**; de aanroeper van een design pattern.	
	client-side	*vakterm*, **server-side** | **GUI**; de (sub)omgeving waar de gebruiker gebruik van kan maken. Hier dient er zo min mogelijk logica gedaan te worden, in tegenstelling tot de server-side.	http://en.wikipedia.org/wiki/Client-side
	compositie	*vakterm*, **UML** | **aggregaat**; in tegenstelling tot de aggregaat, kunnen hierbij beide klassen niet zonder elkaar.	
	cookie	*vakterm*, **HTTP**; een klein databestand dat gebruikt wordt om (tijdelijk) informatie over het/de gebruik(er) op te slaan.	http://tools.ietf.org/html/rfc6265
	CRUD	**Create, Read, Update, Delete**, *vakterm*; een afkorting uit de informatica die staat voor de vier basisoperaties die op duurzame gegevens (meestal een database) uitgevoerd kunnen worden.	http://en.wikipedia.org/wiki/Create,_read,_update_and_delete
D	data contract	*vakterm*, **JSON** | **XML**; een document dat exact aangeeft hoe ontvangen data omgezet moet worden naar een C# object.	http://msdn.microsoft.com/en-us/library/ms733127(v=vs.110).aspx
	design pattern	*vakterm*; een generieke oplossing (qua ontwerp) om een vaak voorkomend probleem op te lossen binnen software.	http://www.objectmentor.com/resources/articles/Principles_and_Patterns.pdf
	DLL	**dynamic-link library**, *vakterm*, **library**; een library met functies, die door meerdere applicaties gebruikt kunnen worden. Een DLL kan op een externe locatie geplaatst worden, waarna (meerdere) applicaties hierna kunnen verwijzen.	http://www.windowsdevcenter.com/pub/a/windows/2005/04/26/create_dll.html?page=1
	draft	*vakterm*; een verzameling documenten(ontwerpen) gepubliceerd door de IETF.	http://www.ietf.org/id-info/
E	epic	*vakterm*, **Scrum** | **user story**; de belangrijkste/hoofd user stories.	
F	façade	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.4.	http://en.wikipedia.org/wiki/Facade_pattern
	form	een formulier om gegevens in te vullen dat opgenomen is in een webpagina en dat door middel van HTML is gecodeerd.	http://www.w3.org/MarkUp/HTMLPlus/htmlplus_41.html
G	GUI	**Graphical User Interface**, *vakterm*, **client-side**; een manier van interactie met een computer doormiddel van grafische beelden en/of tekst	http://en.wikipedia.org/wiki/Graphical_user_interface
H	hash	*vakterm*; data opknippen in vaste lengtes, wordt gebruikt om encryptie toe te passen.	http://en.wikipedia.org/wiki/Hash_function
	HTTP	**Hypertext Transfer Protocol**, *vakterm*; het protocol voor de communicatie tussen een web-client (meestal een webbrowser) en een webserver.	http://tools.ietf.org/html/rfc2616
I	IETF	**Internet Engineering Task Force**; een grote, open, internationale gemeenschap van netwerkontwerpers, - operators, -leveranciers en -onderzoekers die zich bezighoudt met de evolutie van de internetarchitectuur en de soepele werking van het internet.	http://www.ripe.net/internet-coordination/internet-governance/internet-technical-community/ietf
	ISO	**International Organization for Standardization**, *vakterm*; het instituut dat internationale (open) standaarden opstelt en vastlegt. Elk standaard(nummer) wordt in combinatie met de afkorting weergegeven.	http://www.iso.org/iso/home/about.htm
J	JSON	**JavaScript Object Notation**, *vakterm*; zie **Onderzoek Social Media**, hoofdstuk 2.2.2.	http://json-schema.org/examples.html
L	library	*vakterm*, **DLL**; een verzameling code (functies/routines) die door programma's kunnen worden gebruikt. In tegenstelling tot een DLL, wordt met een library (meestal) een statisch gekoppelde library bedoelt. Dit houdt in dat deze ingebouwd moet worden binnen de applicatie.	http://en.wikipedia.org/wiki/Library_(computing)
	localhost	*vakterm*; een loopback-interface die verwijst naar een domein binnen het eigen systeem op het netwerk. Hiermee kan er web-based getest	http://tools.ietf.org/html/rfc2606#section-2

		worden, zonder online te hoeven.	
O	Oauth	**Open Authorization**, *vakterm*; zie **Onderzoek Social Media**, hoofdstuk 2.2.1.	http://developer.yahoo.com/oauth/guide/oauth-auth-flow.html
	ontwikkelteam	*vakterm*, **Scrum**; zie **plan van aanpak**, hoofdstuk 3.1.2.1.2.	
	OO(P)	**Object-oriented programming**, *vakterm*; een paradigma dat gebruikt wordt bij het object georiënteerd programmeren en de objectgeoriënteerde opslag van data.	http://en.wikipedia.org/wiki/Object-oriented_programming
P	parent	*vakterm*, **OO(P)**; de hoofdklasse van een (sub)klasse binnen OO.	
	polymorfisme	*vakterm*; staat voor veelvormigheid. In het geval van de informatica wordt hiermee het volgende bedoeld: het gelijkvormig zijn van de interface van klassen en objecten, maar met verschillende implementaties.	http://en.wikipedia.org/wiki/Polymorphism_(computer_science)
	product-owner	*vakterm*, **Scrum**; zie **plan van aanpak**, hoofdstuk 3.1.2.1.1.	
	proxy	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.5.	http://www.lepus.org.uk/ref/companion/Proxy.xml
	PvA	**Plan van Aanpak**, *vakterm*; een document waarin beschreven staat wat een project inhoudt (reden, stakeholders, werkwijze, planning, etc.)	
R	requirement	*vakterm*; een enkelvoudig gedocumenteerde bepaling, wat een bepaald product of dienst zou moeten doen.	http://en.wikipedia.org/wiki/Requirement
	REST	**Representational state transfer,** *vakterm*; zie **Onderzoek Social Media**, hoofdstuk 2.2.3.	http://www.restapitutorial.com/
S	SaaS	**Software as a Service**, *vakterm*; software dat als een onlinedienst wordt aangeboden.	http://en.wikipedia.org/wiki/Software_as_a_service
	Scrum	vakterm, **software ontwikkelmethode**; zie **plan van aanpak**, hoofdstuk 3.1.2.	https://www.scrum.org/Portals/0/Documents/Scrum%20Guides/2013/Scrum-Guide.pdf
	Scrum-master	*vakterm*, **Scrum**; zie **plan van aanpak**, hoofdstuk 3.1.2.1.3.	
	SDK	**software development kit**, *vakterm*; een verzameling hulpmiddelen die handig zijn bij het ontwikkelen van computerprogramma's voor een bepaald besturingssysteem, type hardware, desktopomgeving of voor het maken van software die een speciale techniek gebruikt.	http://en.wikipedia.org/wiki/Software_development_kit
	server-side	*vakterm*, **client-side**; dit is de omgeving waar zo veel mogelijk logica uitgevoerd moet worden. De omgeving wordt gebruikt door de server computers om de client-side voor de gebruiker op te stellen.	http://en.wikipedia.org/wiki/Server-side
	signature	*vakterm*; digitale handtekening.	
	singleton	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.6.	http://csharpindepth.com/Articles/General/Singleton.aspx
	SmA	**Social media API**, **API**; de API's van social media.	
	SmP	**Social media platform**; bijvoorbeeld Facebook, Google+, LinkedIn, etc.	
	social media	*vakterm*; een verzamelbegrip voor online platformen waar de gebruikers, zonder of met minimale tussenkomst van een professionele redactie, de inhoud verzorgen. Hoofdkenmerken zijn interactie en dialoog tussen de gebruikers.	http://en.wikipedia.org/wiki/Social_media
	SSL	**Secure Sockets Layer**, *vakterm*, **TLS**; een encryptie-protocol die communicatie op het internet beveiligt.	
	strategy	*vakterm*, **design pattern**; Zie **ontwerp**, hoofdstuk 2.3.3.	http://www.lepus.org.uk/ref/companion/Strategy.xml
	String	*vakterm*; een datatype dat uit een reeks tekens of karakters bestaat.	http://msdn.microsoft.com/en-us/library/system.string.aspx
T	template	*vakterm*; een sjabloon voor een stuk code. Met behulp van templates is het mogelijk een stuk broncode te genereren, zodat het voor meer object types bruikbaar is.	http://en.wikipedia.org/wiki/Template_metaprogramming
	TLS	**Transport Layer Security**, *vakterm*, **SSL**; een encryptie-protocol die communicatie op het internet beveiligt.	http://technet.microsoft.com/en-us/library/cc785811.aspx
U	UML	**Unified Modeling Language**, *vakterm*; een modelmatige taal om object georiënteerde analyses en ontwerpen voor een informatiesysteem te kunnen maken.	ISBN: 978-0-321-32127-5
	uni-directioneel	*vakterm*, **UML**; een associatie waarbij één van de klassen weet wat de andere is, maar andersom niet.	
	UNIX-tijd	*vakterm*; een systeem voor het beschrijven van instanties van datum/tijd. De start datum/tijd is 1 januari 1970, 0:00:00h. De waarde van de datum/tijd wordt opgeslagen als één getal.	http://www.idrbt.ac.in/publications/workingpapers/Working%20Paper%20No.%209.pdf
	URL	**Uniform Resource Locator**, *vakterm*; een gestructureerde naam die verwijst naar een stuk data. Wordt vooral binnen het web gebruikt.	http://tools.ietf.org/html/rfc3305
	user story	*vakterm*, **Scrum** | **requirement**; een korte beschrijving (story) van wat een gebruiker (user) wil.	http://guide.agilealliance.org/guide/user-stories.html
X	XML	**Extensible Markup Language**, *vakterm*; een standaard van het World Wide Web Consortium voor de syntaxis van formele opmaaktalen waarmee men gestructureerde gegevens kan weergeven in de vorm van platte tekst.	http://www.w3.org/TR/REC-xml/